BEHAVIOURAL PROFILES

A RELATIONAL APPROACH TO BEHAVIOUR CONSISTENCY

MATTHIAS WEIDLICH

BUSINESS PROCESS TECHNOLOGY
HASSO PLATTNER INSTITUTE, UNIVERSITY OF POTSDAM
POTSDAM, GERMANY

DISSERTATION
ZUR ERLANGUNG DES GRADES EINES
DOKTORS DER NATURWISSENSCHAFTEN
– DR. RER. NAT. –

JULY 2011

Published online at the
Institutional Repository of the University of Potsdam:
URL http://opus.kobv.de/ubp/volltexte/2011/5559/
URN urn:nbn:de:kobv:517-opus-55590
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-55590

Pour L.H.O.O.Q.

ABSTRACT

Business Process Management (BPM) emerged as a means to control, analyse, and optimise business operations. Conceptual models are of central importance for BPM. Most prominently, process models define the behaviour that is performed to achieve a business value. In essence, a process model is a mapping of properties of the original business process to the model, created for a purpose. Different modelling purposes, therefore, result in different models of a business process. Against this background, the misalignment of process models often observed in the field of BPM is no surprise. Even if the same business scenario is considered, models created for strategic decision making differ in content significantly from models created for process automation. Despite their differences, process models that refer to the same business process should be *consistent*, i. e., free of contradictions. Apparently, there is a trade-off between strictness of a notion of consistency and appropriateness of process models serving different purposes. Existing work on consistency analysis builds upon behaviour equivalences and hierarchical refinements between process models. Hence, these approaches are computationally hard and do not offer the flexibility to gradually relax consistency requirements towards a certain setting.

This thesis presents a framework for the analysis of behaviour consistency that takes a fundamentally different approach. As a first step, an alignment between corresponding elements of related process models is constructed. Then, this thesis conducts behavioural analysis grounded on a relational abstraction of the behaviour of a process model, its behavioural profile. Different variants of these profiles are proposed, along with efficient computation techniques for a broad class of process models. Using behavioural profiles, consistency of an alignment between process models is judged by different notions and measures. The consistency measures are also adjusted to assess conformance of process logs that capture the observed execution of a process. Further, this thesis proposes various complementary techniques to support consistency management. It elaborates on how to implement consistent change propagation between process models, addresses the exploration of behavioural commonalities and differences, and proposes a model synthesis for behavioural profiles.

ZUSAMMENFASSUNG

Das Geschäftsprozessmanagement umfasst Methoden zur Steuerung, Analyse sowie Optimierung von Geschäftsprozessen. Es stützt sich auf konzeptionelle Modelle, Prozessmodelle, welche den Ablauf zur Erreichung eines Geschäftszieles beschreiben. Demnach ist ein Prozessmodell eine Abbildung eines Geschäftsprozesses, erstellt hinsichtlich eines Modellierungsziels. Unterschiedliche Modellierungsziele resultieren somit in unterschiedlichen Modellen desselben Prozesses. Beispielsweise unterscheiden sich zwei Modelle erheblich, sofern eines für die strategische Entscheidungsfindung und eines für die Automatisierung erstellt wurde. Trotz der in unterschiedlichen Modellierungszielen begründeten Unterschiede sollten die entsprechenden Modelle konsistent, d.h. frei von Widersprüchen sein. Die Striktheit des Konsistenzbegriffs steht hierbei in Konflikt mit der Eignung der Prozessmodelle für einen bestimmten Zweck. Existierende Ansätze zur Analyse von Verhaltenskonsistenz basieren auf Verhaltensäquivalenzen und nehmen an, dass Prozessmodelle in einer hierarchischen Verfeinerungsrelation stehen. Folglich weisen sie eine hohe Berechnungskomplexität auf und erlauben es nicht, den Konsistenzbegriff graduell für einen bestimmten Anwendungsfalls anzupassen.

Die vorliegende Arbeit stellt einen Ansatz für die Analyse von Verhaltenskonsistenz vor, welcher sich fundamental von existierenden Arbeiten unterscheidet. Zunächst werden korrespondierende Elemente von Prozessmodellen, welche den gleichen Geschäftsprozess darstellen, identifiziert. Auf Basis dieser Korrespondenzen wird ein Ansatz zur Konsistenzanalyse vorgestellt. Jener basiert auf einer relationalen Verhaltensabstraktion, dem Verhaltensprofil eines Prozessmodells. Die Arbeit führt verschiedene Varianten dieses Profils ein und zeigt wie sie für bestimmte Modellklassen effizient berechnet werden. Mithilfe von Verhaltensprofilen werden Konsistenzbegriffe und Konsistenzmaße für die Beurteilung von Korrespondenzen zwischen Prozessmodellen definiert. Weiterhin werden die Konsistenzmaße auch für den Anwendungsfall der Konformität angepasst, welcher sich auf beobachtete Abläufe in Form von Ausführungsdaten bezieht. Darüber hinaus stellt die Arbeit eine Reihe von Methoden vor, welche die Analyse von Verhaltenskonsistenz ergänzen. So werden Lösungen für das konsistente Übertragen von Änderungen eines Modells auf ein anderes, die explorative Analyse von Verhaltensgemeinsamkeiten, sowie eine Modellsynthese für Verhaltensprofile vorgestellt.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

- Matthias Weidlich, Jan Mendling, Mathias Weske: Efficient Consistency Measurement Based on Behavioral Profiles of Process Models. IEEE Transactions on Software Engineering (TSE) 37(3):410-429 (2011)
- Matthias Weidlich, Artem Polyvyanyy, Nirmit Desai, Jan Mendling, Mathias Weske: Process Compliance Analysis based on Behavioural Profiles. Information Systems 36(7): 1009-1025 (2011)
- Matthias Weidlich, Jan Mendling: Perceived Consistency between Process Models. Information Systems. In press. DOI: http://dx.doi.org/10.1016/j.is.2010.12.004 (2011)
- Matthias Weidlich, Artem Polyvyanyy, Jan Mendling, Mathias Weske: Causal Behavioural Profiles - Efficient Computation, Applications, and Evaluation. Fundamenta Informaticae. In press (2011)
- Sergey Smirnov, Matthias Weidlich, Jan Mendling: Business Process Model Abstraction based on Synthesis from Well-Structured Behavioral Profiles. Technical Report. Hasso Plattner Institute (2011)
- Matthias Weidlich, Jan Mendling, Mathias Weske: A Foundational Approach for Managing Process Variability. CAiSE 2011:267-282
- Matthias Weidlich, Remco M. Dijkman, Jan Mendling: The ICoP Framework: Identification of Correspondences between Process Models. CAiSE 2010:483-498
- Matthias Weidlich, Artem Polyvyanyy, Nirmit Desai, Jan Mendling: Process Compliance Measurement Based on Behavioural Profiles. CAiSE 2010:499-514
- Sergey Smirnov, Matthias Weidlich, Jan Mendling: Business Process Model Abstraction Based on Behavioral Profiles. ICSOC 2010:1-16
- Matthias Weidlich, Artem Polyvyanyy, Jan Mendling, Mathias Weske: Efficient Computation of Causal Behavioural Profiles Using Structural Decomposition. Petri Nets 2010:63-83
- Matthias Weidlich, Felix Elliger, Mathias Weske: Generalised Computation of Behavioural Profiles Based on Petri-Net Unfoldings. WS-FM 2010:101-115
- Matthias Weidlich, Mathias Weske: Structural and behavioural commonalities of process variants. ZEUS 2010:41-48

- Matthias Weidlich, Mathias Weske: On the behavioural dimension of correspondences between process models. ZEUS 2010:65-72
- Matthias Weidlich, Mathias Weske, Jan Mendling: Change Propagation in Process Models Using Behavioural Profiles. IEEE SCC 2009:33-40
- Matthias Weidlich, Alistair P. Barros, Jan Mendling, Mathias Weske: Vertical Alignment of Process Models - How Can We Get There?. BMMDS/EMMSAD 2009:71-84

it's the ending of a play
and soon begins another
hear the leaves applaud the wind

— Emiliana Torrini

ACKNOWLEDGMENTS

Looking back, it is evident that a number of people have a significant stake in this work. At this point, I would like to thank them all for their guidance, support, and inspiration.

I am grateful to Mathias Weske, my supervisor, who supported me in any respect during my PhD. His continuous encouragement and the offered freedom for my research have been essential for me to complete this work, and to enjoy it. I owe much to Jan Mendling, who virtually became a second supervisor of mine. His insightful feedback along with his positive spirit pushed me forward more than once. There are many more people to thank for their support, in the one way or the other. Alistair Barros convinced me that consistency matters, which has been the starting point for this thesis. Collaborating with Remco Dijkmann led to substantial progress in model matching, thank you. Joint work on model understanding with Dirk Fahland, Jakob Pinggera, Hajo Reijers, Barbara Weber, and Stefan Zugal was beneficial to see my research from a different perspective. Also, I am grateful to Wolfgang Reisig and Wil van der Aalst, my reviewers, for their valuable feedback, especially with respect to the positioning of my work.

A big thank you is due to all my colleagues and former colleagues from the Business Process Technology group for plenty of inspiring discussions, stimulating collaborations, and a pleasant work environment. Alex was a great office mate throughout the last years, always eager to discuss scientific and not-so scientific topics. I really enjoyed diving into the world of temporal logics with Ahmed, fighting for formal results all evening long with Artem, discussing language formalisations with Gero, inventing fancy metrics with Matthias, and reaching higher levels of abstraction with Sergey. Thanks heaps for that!

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

ALIGNMENTS OF PROCESS MODELS

1 INTRODUCTION

All models are wrong;
some models are useful.
— George E. P. Box [62]

Business is process-driven. Since process orientation emerged as an organisation principle [165, 100, 195, 406], the last decades have seen a remarkable uptake of Business Process Management (BPM). BPM has been established as a means to control, analyse, and optimise business operations. This trend is observed independent of any business domain and organisational background.

Conceptual models are at the core of BPM. Among them, process models describe the behaviour that is performed to achieve a business value [513]. Pragmatics is an inherent feature of every conceptual model. Mapping and reducing the reality can be seen as the elementary steps of model creation [251]. The *purpose* of a model answers the question of what to map and what to reduce. Against this background, the misalignment of process models often observed is no surprise. Even if the same business scenario is considered, models created for strategic decision making differ in content significantly from models created for process automation. The well-known 'Business-IT-Gap' [70, 186, 388] is only the most prominent incarnation of this problem. The question of how to assess consistency between behavioural models is fundamental.

In this chapter, we approach this question by reviewing the essentials of model creation in Section 1.1. Then, we focus on drivers for the creation of process models in Section 1.2 and the need to assess behaviour consistency between them in Section 1.3. This gives rise to the characterisation of the problem addressed by this thesis in Section 1.4. Section 1.5 summarises our contributions. Finally, we outline the structure of this thesis in Section 1.6.

1.1 THE ESSENCE OF MODELLING

Abstraction is an essential task in Computer Science. Models are created to cope with the complexity of real-world phenomena

and to establish a well-defined universe of discourse. Aho and Ullman deduce a characterisation of Computer Science from the notion of abstraction.

> *'Computer Science is a science of abstraction, creating the right model for a problem and devising the appropriate mechanizable techniques to solve it.'*
>
> — Alfred V. Aho and Jeffrey D. Ullman [14]

Despite their importance, well-established notions of a model and an abstraction are missing even in a rather narrow domain, such as model-driven engineering [251, 208, 252]. The question of what constitutes a model has been debated extensively, see [251, 208, 252, 53, 415, 289, 52]. The following three features that are rooted in the model theory by Stachowiak [430] are commonly adopted for models in Computer Science, cf., [251, 289].

Mapping feature. A model is grounded on an original, may it be an object or a phenomenon. The original is mapped to the model. The original may be non-existent. It may be planned, suspected, or fictitious [289].

Reduction feature. A model is a reduced representation of the original. Only a selection of properties of the original is mapped to the model.

Pragmatics feature. With respect to a certain purpose, the model can be used as a replacement for the original. In other words, the model is created for a dedicated purpose.

These features give rise to a generic characterisation of abstraction as the act of model creation. That is, abstraction is a projection applied to an original. The projection reduces the amount of information by filtering properties of the original [251, 227]. The question of what to project, in turn, is answered by the purpose of the model, i. e., its pragmatic feature. The purpose guides the selection and granularity of properties of the original that should be contained in the model. Hence, the quality of a model is determined by its ability to answer certain questions regarding the original [278]. If the model is adequate, any deviation between the answers obtained from the model and those given by the original are of an acceptable extent [53, 251]. A model may be imprecise or even incorrect with respect to certain properties of the original. Still, it may be adequate, if conclusions drawn on the original are valid within the required level of confidence.

The notions of a model and an abstraction are illustrated by Figure 1. Taking the example of a space shuttle orbiter, the original is referred to by the upper picture in Figure 1. Below, there are two models of the orbiter. Both are derived by abstraction

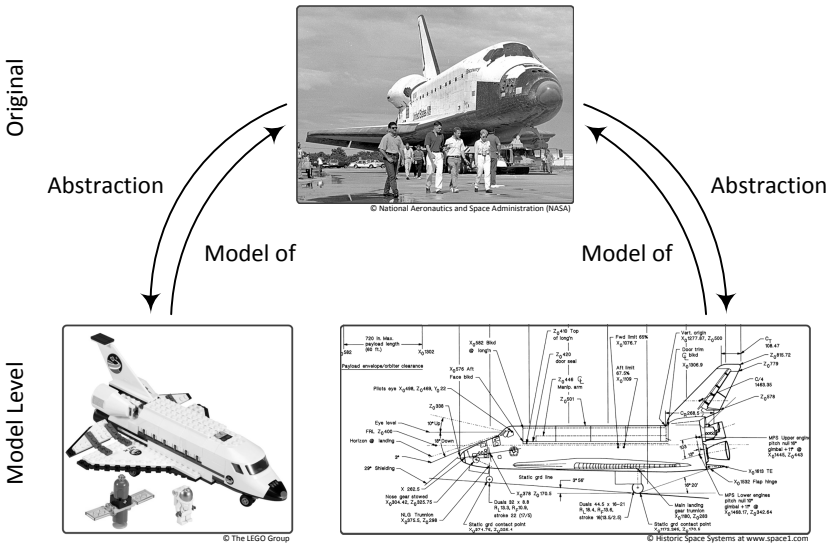


Figure 1: The essence of modelling.

from the original object. Apparently, both models provide a simplistic view on the original. Innumerable properties of the original have not been mapped to the models. Nevertheless, both models are adequate regarding a certain purpose. The toy model of the orbiter in the lower left corner of Figure 1, for instance, allows for moving the ailerons at the wings. Mapping this property from the original to the model has been considered to be relevant for a toy model. The construction plan in the lower right corner of Figure 1 has been created for a different purpose. It allows for drawing conclusions on the spatial dimensions of the orbiter. Besides its simplicity, this example illustrates the importance of pragmatics when creating a model from an original object or phenomenon by abstraction.

1.2 DRIVERS OF PROCESS MODELLING

Process orientation is an organisational principle that has its roots in business administration [165, 406] and organisational redesign [100, 195]. It emphasises business processes – a collection of activities performed in coordination to realize a business goal – as the source of value creation. Business Process Management (BPM) comprises means to support the design, administration, configuration, enactment, and analysis of business processes [513]. BPM relies on explicit representations of business processes in order to implement these operations. To this end, process models are commonly used. They describe a collection of activity models along with their logical and temporal order [40, 513].

Following the discussion of abstraction in the previous section, process models are an abstraction of business operations,

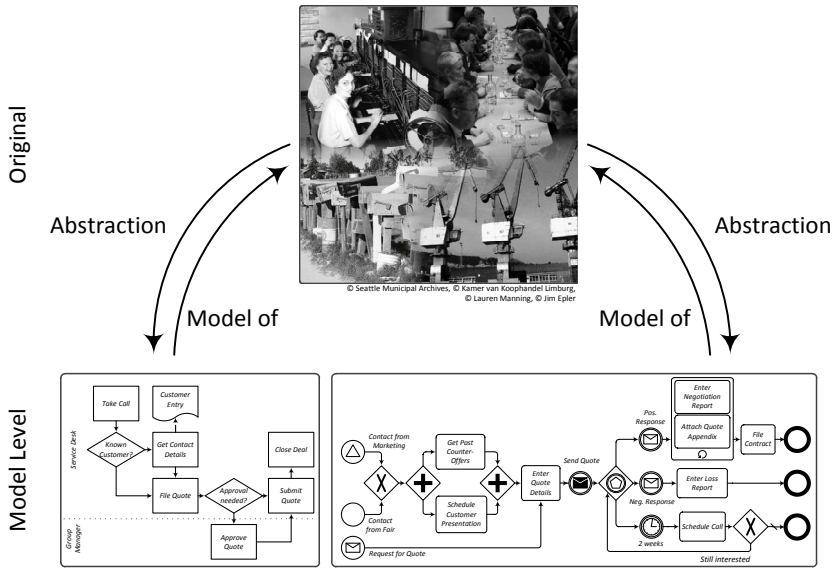


Figure 2: The essence of process modelling.

cf., Figure 2. There is a mapping of the essential properties of a business process, which may be planned, suspected, or fictitious, to a process model. Further, process models are a reduced representation of a business process. Activities are captured at a certain level of granularity and contextual information such as involved roles, data artefacts, or information systems may be explicitly selected to be captured in the process model. Again, we observe the pragmatic feature of a model. The purpose of the process model answers the question of what to map and what to reduce when capturing business processes.

Process models may be created for manifold purposes. In the remainder of this section, we review applications of process models. This overview does not aim at completeness or the identification of orthogonal application areas. Instead, it shall illustrate the spectrum of potential applications of process modelling by some prominent examples.

Process Understanding & Communication. Process models are created to document the way business operations are conducted. The aim is to arrive at a consistent understanding of these operations and improve the communication of business processes among different stakeholders [513, 40]. Collaboration within a large organisation requires a common understanding of the business processes. Such a common understanding is hindered by the different backgrounds of process stakeholders. From the field of data schema modelling, it is known that *'different user groups or designers adopt their own viewpoints in modelling the same objects in the application domain'* [34]. Process models are used as a means to create a shared understanding and aim at bridging the local

views of particular stakeholders on business processes. In addition, process models may also be used in the design of organisational positions or for employee training. We summarise that process models created for this purpose can be seen as artefacts for the knowledge management within an organisation. As indicated by a recent Delphi study [218], improved understanding and communication are among the most prominent perceived benefits of process modelling.

Process Improvement. Process modelling is frequently motivated by process improvement [100, 200]. According to the aforementioned study [218], it is perceived to be the top benefit of process modelling in practise. There is a large spectrum of criteria that may be targeted when improving business processes, such as cost or cycle-time reduction, or increased quality of products or services. Process models are used at different stages of process improvement initiatives. For instance, they are leveraged for measuring the performance of business processes to identify bottlenecks or quality problems. Further, process redesign efforts are guided by process models depicting the as-is state and those that capture the to-be state of a business process [377].

Process Simulation. To forecast the impact of changes in internal or external parameters on business operations, business processes are simulated [296, 403]. Process models that depict the current or intended business processes are annotated with simulation relevant data, such as execution costs, execution times, process instantiation frequencies, or the availability of resources. Then, process simulation allows conclusions to be drawn on the performance of a business process before it is implemented or changed.

Process Automation. Process models are used as blueprints for the design of process-aware information systems [134]. To this end, process models capture business requirements that have to be met by the supporting IT-infrastructure. As such, they are intended to bridge the gap between business requirements and system specifications [70, 186, 388]. Process models are also used for process automation using workflow technology [270, 439]. Workflow engines take a process description as input and execute it by enforcing the predefined behaviour. Hence, a process model may define a concrete technical orchestration of services realising the business process.

Process Certification & Process Compliance. Certification and compliance analysis of business operations is often conducted in a process-oriented way. Various certification initiatives eval-

uate an organisation based on its processes. As an example, the ISO-9001 standard[1] relates to the quality management system of an organisation and explicitly requires the documentation of business processes. Although not required for certification, the ISO-9001 recommends capturing business processes by process models. Recently, techniques emerged to conduct process-oriented risk management [237, 537] and compliance analysis [384, 3, 27]. Process models are used to structure risk analysis and to check whether operations adhere to policies, internal directives, and external regulations. For instance, the Control Objectives for Information and related Technology (COBIT) framework[2] aims at managing risks and adhering to compliance demands using a process-oriented approach.

Business to Business (B2B) Integration. In recent years, the focus of BPM shifted from a single organisation to cross-organisational integration, driven by information systems. Organisations outsource single business activities or even complete business processes, which results in distributed value chains that need to be coordinated [78]. Such coordination is typically process-driven. Process models are used to define the interface and the protocol followed by different organisations as part of their collaboration. For instance, the RosettaNet[3] consortium standardised business processes between trading partners in the electronic commerce sector.

As stated above, this overview of drivers for process modelling cannot be complete. Still, it illustrates that there is a whole spectrum of drivers for process modelling, from employee training to B2B integration. The variety of purposes for process modelling can be assumed to have a dramatic influence on the way process models are created.

1.3 DRIVERS OF CONSISTENCY ANALYSIS

The drivers of process modelling reviewed in the previous section cannot be organised in a strict top-down fashion. Hence, it is unrealistic to assume that the corresponding process models can always be derived through hierarchical refinement. Consequently, and most likely, there will be a variety of differences between models. Depending on the purpose of model creation, there is a huge difference in the appropriate level of abstraction of a business process, as well as the assumed perspective. As

1 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=46486
2 http://www.isaca.org/cobit
3 http://www.rosettanet.org/

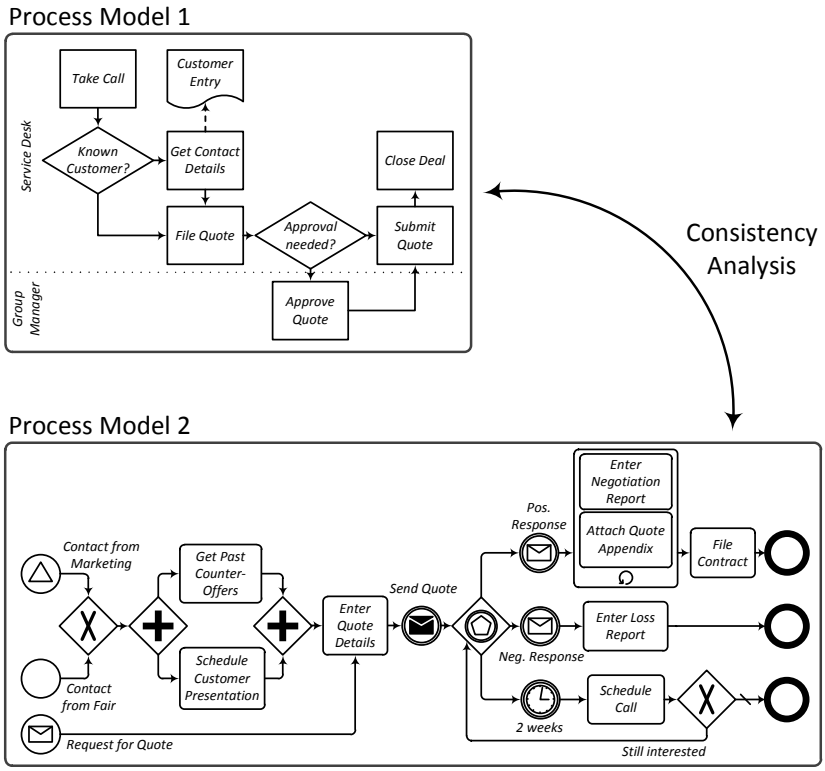


Figure 3: Consistency analysis of process models.

discussed for models in general, every process model has to be *appropriate* – it must incorporate a reasonable level of detail, focus on certain properties of a business process, and neglect irrelevant aspects. Arguably, *mismatches*, i. e., differences in the model structure or the specified behaviour, are in the nature of process models that serve different purposes. Avoidance of such mismatches may not only be impossible, it may also be unnatural and counter-productive. That is to say that a resolution of these mismatches may impact the adequacy of a process model in a negative manner.

We focus on the purpose of modelling as the source of differences between process models that represent (overlapping parts of) the same business process. However, differences may also stem from other factors, e. g., related to the act of model creation. The modelling methodology followed or the expressiveness of the chosen modelling language may cause differences between related process models, see [31] for factors that influence the modelling process. Nevertheless, we emphasise that even controlling all of these factors will not avoid the aforementioned mismatches if process models are created for different purposes.

We illustrate differences between related process models with a simple lead-to-order process. In essence, this process involves establishing contact with a customer, submitting a quote, and handling the customer's response. The process models in Fig-

ure 3 illustrate that the way this process is represented differs with respect to the modelling purpose. The upper model may be used to illustrate the overall processing and to clarify organisational responsibilities. This model focusses on the major activities and decision points of the process. The lower model provides a more fine-grained view of the operations. It aims at capturing technical aspects, such as different instantiation scenarios and the treatment of exceptional cases. This model may be used to implement and configure supporting information systems.

Granted that there are multiple process models that capture the same business process, their consistency has to be analysed. The need to have consistent representations of business operations is inherent. Imagine that process models used as means of communication are inconsistent with process models created for process automation. Then, the benefit of creating a shared understanding among process stakeholders with the former model is of limited use, as it deviates from the way operations are supported with information systems. We concretise drivers for consistency analysis with the following use cases.

Validation. One process model is utilised as a specification. A second, often more fine-grained model is validated against the specification. This scenario is observed in the context of business-centred organisational process models and implementation-centred technical process models, see Figure 3. However, validation is not restricted to such a setting. A different example would be the validation of a model that captures the non-technical process implementation in a certain organisational environment against a model representing compliance requirements that have to be obeyed.

Inter-Model Analysis. Process optimisation often relies on an analysis across multiple process models. Aspects that are captured in different models have to be related to each other to draw conclusions on the overall processing. For the scenario depicted in Figure 3, information on the actual processing (e. g., processing times) obtained for the lower model, may be related to the roles defined in the upper model.

Change Propagation. Business processes and, therefore, process models continuously undergo changes. To keep different representations of business operations in sync, changes applied to one process model may require updating all related models accordingly. Against the background of process models assuming different abstraction levels and perspectives, automatic change propagation appears to be unrealistic. Still, the identification of process models or process model regions that are affected by a change would be a major benefit.

Addressing these use cases requires means for correlating elements of different process models. Such correspondences are established between process model elements, or sets thereof. They have to respect certain consistency criteria to be exploited for model validation, analysis, or change propagation.

1.4 PROBLEM STATEMENT

This thesis takes the diverging drivers of process modelling and the need to assess the consistency of related process models as a starting point. Our focus is on the control flow perspective. Even though state-of-art process modelling goes beyond pure control flow specification, process models, first and foremost, are behavioural models. The definition of a coordination of activity execution is the very core of a process model. Hence, it is reasonable to approach consistency analysis from the control flow perspective before taking additional perspectives, e. g., data flow modelling or resource assignments, into account. Further, we concentrate on process models that are defined following a procedural modelling paradigm. Recently, approaches to process modelling that break with this paradigm have emerged, e. g., declarative [452, 462, 327] or artefact-centric process modelling [89, 54]. To date, there has not been any significant uptake of these approaches in practise [102, 210], though.

We focus on process models that represent the same business process. Deciding whether two process models refer to the same business process may not be straight-forward. The question of what constitutes a business process – when is it initialised and when does it complete? – is influenced, among other factors, by the drivers of process modelling. Hence, it is likely that related process models are only partially overlapping in terms of their coverage of business operations. However, this aspect is of secondary importance for our work. When referring to process models that capture the same business process, therefore, we do not assume a unique characterisation of initialisation and completion of the business process.

We phrase the research question of this thesis as follows.

> How to assess behaviour consistency for process models capturing the same business process?

According to Zelewski, consistency of process models refers to a *freedom of contradictions* [533]. Still, a concrete operationalisation of this definition remains challenging. Evidently, there is a trade-off between *strictness* of a consistency notion and *appropriateness of process models* serving different purposes. A strict notion, which requires all information of one model to be present

in another model as well, will result in models that are inappropriately tailored for their different purposes.

The manifold drivers for process modelling and consistency analysis suggest that any criterion to assess behaviour consistency must be defined in a relativistic manner. A single criterion cannot satisfy the requirements for behaviour consistency in all contexts. Such a relativistic angle is also acknowledged in the literature on behaviour equivalences.

> *'When semantic equivalences are used in the design of concurrent systems, or for verification purposes, they should be chosen in such a way that two system descriptions are considered equivalent only if the described behaviours share the properties that are essential in the context in which the system will be embedded. It depends on this context and on the interests of a particular user which properties are essential.'*
>
> — Robert J. van Glabbeek [475]

It is remarkable that the relevance of the context and the interests of particular users, i. e., the purpose of applying an equivalence notion, are emphasised even for the design and verification of concurrent systems. In system design, a set of models jointly specifies the system to be built. One would assume that the consistency requirements are much stricter than for the case of process models serving different purposes. Therefore, the analysis of related process models, a fortiori, requires a relativistic assessment of behaviour consistency.

To narrow the scope of this thesis, we elaborate on requirements that shall be met when answering the research question.

Flexible Assessment. Behaviour consistency between related process models has to take a relativistic angle. There have to be means to adapt the consistency criterion towards a concrete setting or to interpret it against the background of the considered models. A single Boolean criterion cannot be assumed to be suited for all of the aforementioned contexts. Instead, fine-granular criteria are needed.

Meaningful Feedback. Closely related to flexibility in the consistency assessment is the need to provide meaningful feedback. Given a consistency criterion, traceability of the consistency assessment must be ensured. Sources of inconsistency have to be identified and isolated to be able to interpret the consistency result.

Efficient Computation. Change is the rule not the exception when managing business operations. Business processes are continuously changing. Hence, models representing these processes are also changing frequently, which impacts on the behaviour consistency between them. A consistency criterion

that is computed efficiently supports these changes. Consistency analysis may be conducted in an online fashion and helps to guide the adaptation of process models.

1.5 CONTRIBUTIONS

To answer the research question formulated in the previous section, this thesis makes the following contributions.

(1) Identification of Complex Correspondences

Analysis of behaviour consistency of related process models requires the identification of corresponding model elements. This problem closely relates to a large body of work on data schema and ontology matching. The identification of complex correspondences – not single elements but sets of elements correspond to each other – has been largely neglected in this research area, though. For process models, there has not been any work in this direction. Our contribution is a framework that defines a system architecture for the definition of matchers to derive complex correspondences between two process models. Besides the architecture, we introduce a set of basic matching components used to assemble such matchers.

(2) Definition of Behavioural Profiles

Our approach to behaviour consistency is based on an abstraction of the behaviour of a process model, the behavioural profile of the model. Such a profile captures behavioural characteristics by relations between pairs of activities. Our contribution is the definition of different variants of these profiles for a generic behavioural model, i. e., net systems. Hence, behavioural profiles induce a set of relational semantics for behavioural models.

(3) Computation of Behavioural Profiles

The abstraction of a behavioural profile can be computed efficiently for a broad class of process models. For sound free-choice WF-systems, we present formal results that allow for deriving behavioural profiles from the model structure in low polynomial time. We also show how structural decomposition techniques are applied to speed up the computation and support partial computation of behavioural profiles. These results are complemented by an approach to the computation of behavioural profiles from a complete prefix unfolding, which is computationally hard but applicable in a more general case. Hence, our contribu-

tion is a set of techniques for the computation of behavioural profiles tailored towards dedicated classes of process models. We also present an extensive experimental evaluation of these techniques using three model collections from industry.

(4) Consistency Analysis of Process Models

Using the notion of a behavioural profile, we present a framework for the analysis of behaviour consistency of process models. Starting with different Boolean notions of behaviour consistency, we elaborate on how to quantify behaviour consistency, how to support behaviour consistent change propagation between process models, and how to explore behavioural commonalities and inconsistencies. As part of that, we present a set algebra for behavioural profiles and an approach to model synthesis from behavioural profiles. Hence, our contribution is a set of integrated techniques to manage behaviour consistency in a holistic way. Our consistency criteria have been validated empirically against the consistency perception of process modelling experts. Further concepts have been evaluated in a series of experiments using process models from industry.

(5) Consistency Analysis of Process Logs

All of the above contributions consider behaviour consistency on the level of process models. Any conclusions drawn from these results neglect the way business operations are actually conducted, i. e., how well a process model represents these operations. To take this aspect into account when interpreting consistency results between process models, we show how to assess consistency, aka conformance, of process logs that capture the observed execution of a process. This approach is grounded on the same formalism used to assess process model consistency – we leverage behavioural profiles for conformance analysis. We also report on findings from an evaluation of this approach in an industrial case study.

1.6 STRUCTURE OF THIS THESIS

We conclude this chapter with an overview of the structure of this thesis. Figure 4 illustrates the structure. The thesis consists of three parts. Although the order of parts and chapters follows upon the procedure to assess behaviour consistency, certain chapters may be skipped in a first reading. We clarify dependencies between chapters in the remainder of this section.

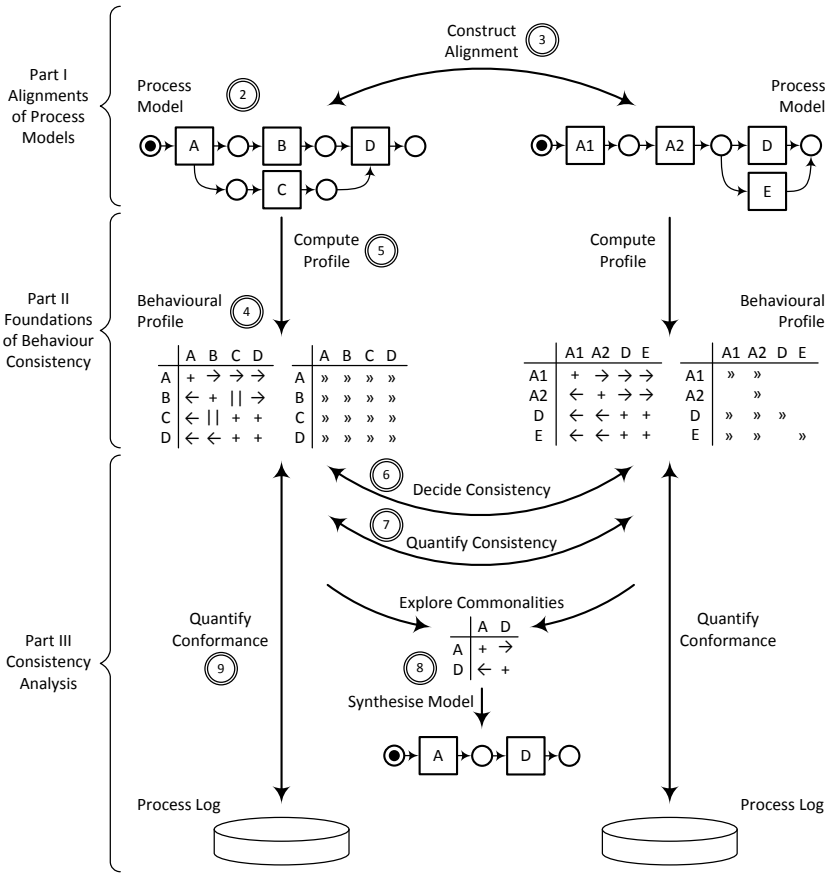


Figure 4: Structure of this thesis. Encircled numbers relate to chapters.

Part I : Alignments of Process Models

The first part focusses on preliminaries for any kind of behaviour consistency analysis. Chapter 2 gives on overview of common process description languages and introduces the formal framework for our work, i. e., the formalism of net systems. Readers familiar with these languages and basic notations of net systems may skip this chapter.

Chapter 3 deals with the construction of an alignment, the identification of correspondences between process models. We clarify terminology, review literature on model matching, and introduce the ICoP framework for the identification of complex correspondences between process models. Assuming an intuitive understanding of the concept of a correspondence relation, readers interested only in the behavioural analysis may skip this chapter.

Part II : Foundations of Behaviour Consistency

The second part introduces the foundations for our approach to behaviour consistency. Chapter 4 presents behavioural pro-

files as an abstraction of the behaviour defined by a process model. As such, this chapter provides the basis for all remaining chapters. In Chapter 5, we introduce techniques for the computation of behavioural profiles. The separation of the definition of behavioural profiles in Chapter 4 from the computation techniques in Chapter 5 allows the reader to skip the formal groundings of the derivation of behavioural profiles. In contrast to the definitions in Chapter 4, the techniques proposed in Chapter 5 are not required to understand the analysis introduced in the third part of this thesis.

The description of behavioural profiles and their computation in the second part is largely independent of our use case of behavioural analysis of related process models. Behavioural profiles have also been leveraged for further use cases, which we briefly discuss in the conclusions of this thesis. Hence, the foundations laid by Chapter 4 and Chapter 5 are independent of any use case for the application of behavioural profiles.

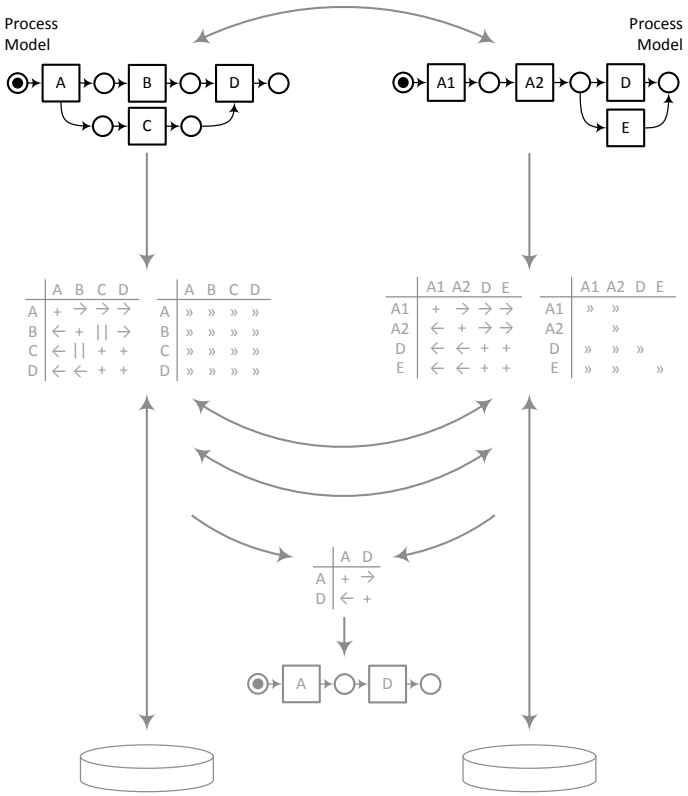
Part III : Consistency Analysis

The third part focusses on the actual consistency analysis based on behavioural profiles. First, Chapter 6 proposes different notions of behaviour consistency that are based on behavioural profiles. We also present an empirical evaluation of these notions with respect to the consistency perception of process modelling experts. Chapter 7 extends these results by focussing on the quantification of behaviour consistency. We propose measures to assess the quality of an alignment and elaborate on techniques to propagate changes between aligned process models that aim at preserving the quality of the alignment. Once the alignment quality has been assessed, further analysis on the behavioural commonalities and differences is addressed in Chapter 8. We introduce a set algebra for behavioural profiles to compute with behaviour and elaborate on the synthesis of a process model from a behavioural profile. Chapter 9 turns the focus on conformance between a process model and the observed execution of a process. We introduce conformance measures based on behavioural profiles and elaborate on techniques for the identification of root causes of non-conformance.

Finally, Chapter 10 concludes this thesis. We give a summary of our results and discuss their relevance in a broader context. Further, we reflect on limitations and give an outlook on directions for future research.

2

PROCESS MODELS



Process models are explicit representations of business processes. On an abstract level, a process model describes a collection of activity models along with their logical and temporal order [40, 513]. The act of creating a process model is influenced by a modelling technique. Following the terminology of [312], a modelling technique comprises a modelling language and a modelling method. The former provides a syntax (also called grammar), semantics, and a notation for the creation of process models, cf., [485, 199]. The latter defines a methodological procedure in which the modelling language is used [485].

In this chapter, we review basic modelling languages and introduce the formal framework used throughout this thesis. In Section 2.1, we focus on process description languages commonly used in practise. A complete discussion of these languages is beyond the scope of our work. However, by introducing four exemplary process description languages, we explicate commonalities and differences in their syntax, semantics, and notations.

In Section 2.2, we turn the focus on Petri net systems, a formalism for the definition of behaviour. Since this formalism well-investigated and conceptually close to common process description languages, it is often used as the basis for behavioural analysis of process models. All results in the remainder of this thesis are obtained for and presented with net systems. We elaborate on the link between common process description languages and net systems in Section 2.3. We conclude this chapter with a discussion in Section 2.4.

2.1 PROCESS DESCRIPTION LANGUAGES

We review four process description languages frequently used in practise, the Business Process Model and Notation (BPMN), Event-Driven Process Chains (EPCs), UML Activity Diagrams (UML ADs), and the Web Service Business Process Execution Language (BPEL). There exists a multitude of process description languages, so that we cannot provide an exhaustive overview. We selected the aforementioned languages since they are widely used [102, 210], and nicely illustrate certain differences that stem from their primary field of application. For instance, EPCs have their roots in business administration, whereas UML ADs fit well into the UML framework with a focus on software engineering. Hence, the usage of one language or the other is closely related to the purpose of process modelling, cf., Section 1.2. All discussed languages have in common that they follow a procedural modelling paradigm. This paradigm has been questioned for certain scenarios, which led to alternative approaches to process modelling, such as declarative process modelling languages [452, 462, 327] or artefact-centric modelling approaches [89, 54]. Since these approaches have not yet seen a remarkable uptake in practise [102, 210], we focus on procedural process description languages.

Business Process Model and Notation

The Business Process Model and Notation (BPMN) is a widely adopted standard for modelling business processes. It has been proposed by the Business Process Management Initiative (BPMI) and was later standardised by the Object Management Group (OMG)[1]. Recently, OMG published the version 2.0 of the language [2].

BPMN 2.0 introduces different types of process diagrams to model business processes. High-level interactions between organisations are captured using *conversation diagrams* that define

1 http://www.omg.org/

communications, a collection of message exchanges, between participants. Interactions are modelled in detail using *choreography diagrams*, which specify a protocol as the process of message exchanges. Finally, the internal business processes of an organisation are modelled using *collaboration diagrams*. We focus on this type of model. It can be seen as the core of BPMN to capture business processes. The diagrams that assume an inter-organisational view have been presented only recently.

Collaboration diagrams provide a comprehensive set of elements to model business processes. A step as part of business operations is captured by an activity. It may be atomic, then called a task, or hierarchically structured, then referred to as a subprocess. Further, BPMN introduces event constructs to model the occurrence of real-world events. Those are classified along two dimensions, their event type and their trigger. The former relates to their role in the process, e. g., to start a process or to interrupt a subprocess. The latter refers to the cause of event occurrence, e. g., the reception of a message or the expiration of a timer. BPMN also introduces pools and lanes to associate activities to roles, and data objects to illustrate data flow in the process.

A BPMN collaboration diagram has a graph structure. The logical and temporal order between activities and events is modelled using sequence flows. Control flow routing that goes beyond sequencing is implemented by explicit gateways, which split and join the control flow. Depending on its type, a gateway realises exclusive disjunctive (XOR), inclusive disjunctive (OR), or conjunctive (AND) semantics. Support for more advanced routing behaviour is achieved by complex and event-based gateways. Further, BPMN control flow routing for exception handling and compensation is grounded on event-based mechanisms.

With version 2.0, BPMN provides a complete, albeit informal, characterisation of execution semantics for all model elements. This characterisation is inspired by Petri nets and assumes a notion of token flow. In previous versions of BPMN, there have been issues in the definition of execution semantics, which led to various formalisation efforts [125, 526, 57, 484]. This work partly influenced the definition of execution semantics for BPMN 2.0, which for instance, adopts the semantics proposed in [484] for the converging OR gateway. Besides execution semantics, the BPMN 2.0 specification also defines a serialisation format and clarifies the way data and service bindings are established. Thus, BPMN collaboration diagrams may be defined such that the process is directly executed using a workflow engine.

An exhaustive discussion of BPMN modelling elements and the language's capabilities is beyond the scope of our work. For introductions to BPMN, we refer the reader to [513, 515, 18, 164].

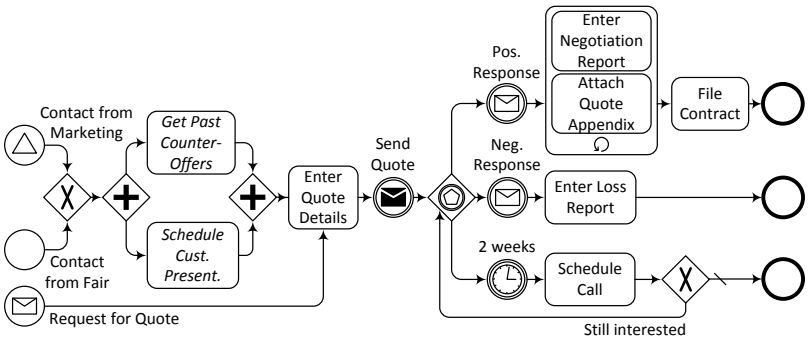


Figure 5: A model of a lead-to-order process depicted as a BPMN collaboration diagram.

Also, the *Berliner BPM-Offensive* published an overview of the notational elements as a poster[2] that is publicly available.

We illustrate the notation used for BPMN collaboration diagrams with an example. Figure 5 depicts a lead-to-order process. In essence, this process involves establishing contact with a customer, submitting a quote, and handling the response from a potential customer. BPMN activities are depicted by rounded rectangles, events are modelled by circles potentially with a symbol highlighting the event trigger, and diamond shapes represent gateways. For the latter, a cross represents exclusive disjunctive behaviour, a plus sign represents conjunctive behaviour. Figure 5 also illustrates some advanced BPMN concepts. For instance, the gateway represented by a diamond that comprises a pentagon models an event-based decision. It realises the *deferred choice* pattern, cf., [456]. At this point, the process reacts to an external decision – the customer replies positively or negatively, or there is a time-out.

Event-Driven Process Chains

Event-Driven Process Chains (EPCs) [231, 336] are another popular notation for modelling business processes. They are often used to capture process models for communication. EPCs are one puzzle piece in the Architecture of Integrated Information Systems (ARIS) framework [405] for process management, these days promoted by Software AG[3]. This framework provides means to integrate organisational, functional, data, and service modelling. The different dimensions are glued together by process models in EPC notation.

EPC process models are a graph comprising functions and events in alternating order. Functions describe elementary business actions. Events capture the process state and, therefore,

2 http://www.bpmb.de/poster
3 http://www.softwareag.com/aris/

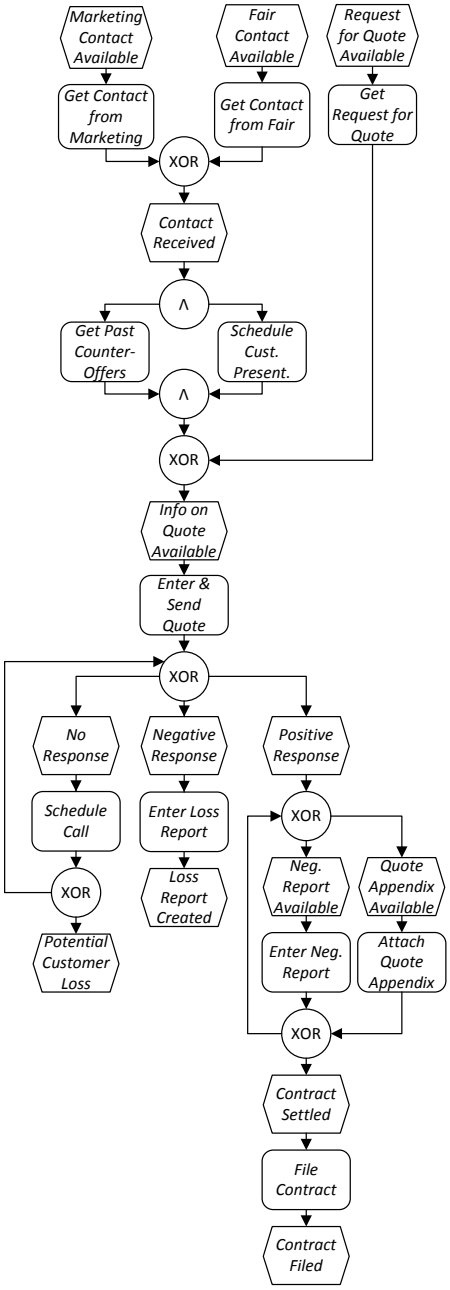


Figure 6: A model of a lead-to-order process depicted as an Event-Driven Process Chain.

define the state before or after a function is executed. Control flow dependencies are expressed using directed flow edges. Similar to BPMN, control flow routing is realised by a dedicated type of nodes, split and join connectors. Those are typed to represent an exclusive disjunction (XOR), an inclusive disjunction (OR), or a conjunction (AND). The alternating order of functions and events induces certain syntax rules that have to be respected when building an EPC graph. For instance, an XOR split must

not be followed by functions, but by events. A formal definition of EPC syntax can be found in [313, 312].

To realise the integration with the organisational, functional, data, and service modelling within the ARIS framework, extended EPCs (eEPCs) allow for annotating functions with additional information, such as organisational units, data objects, and information systems. Extended EPCs also introduce process interfaces to structure a process model hierarchically.

There exist various formalisations of execution semantics for EPCs [116, 236, 472, 312]. As for BPMN, semantics of the converging OR connector have been in the centre of interest. A comparison of EPC semantics can be found in [312]. Further, instantiation semantics of EPCs may not be well defined if there are multiple events without predecessors. These issues have been investigated in detail in [109], a solution to identify sound instantiation scenarios is described in [361].

We illustrate the EPC notation with the example depicted in Figure 6. Functions are denoted by a rectangular shape, events are visualised by hexagon shapes, and connectors are modelled by circles that contain a marker indicating the connector type. Again, the model captures a lead-to-order process. Even though the modelled process is similar to the one captured in Figure 5 using BPMN, both models are not semantically equivalent. The deferred choice pattern can only be approximated with an EPC model. For the sake of simplicity, there are also differences in how both models capture the negotiation phase.

The example illustrates that the basic constructs of EPCs have corresponding elements in BPMN. This allows for transformations between models that use a shared set of concepts in many cases. However, both languages also show syntactical differences, e. g., EPCs are bipartite graphs, and subtle semantic differences, e. g., EPC events may represent states, whereas BPMN events are associated to triggers. Transformation challenges that stem from these differences have been investigated in [216, 114].

UML Activity Diagrams

The Unified Modeling Language (UML) [1] is a framework for multi-perspective modelling in the context of object oriented software engineering. Standardised by the Object Management Group (OMG)[4], UML is widely recognised and established for the design of software systems. It is used in conjunction with many software development methods and can be seen as the de-facto standard in industry.

UML introduces 14 different types of models to support the development of a software system. Structure diagrams, such as

4 http://www.omg.org/

class diagrams, component diagrams, or object diagrams, are used to model the static structure of a system – the system architecture. Behaviour diagrams, in turn, capture the system behaviour – the functionality of a system. The spectrum of behaviour diagrams ranges from use case diagrams that model the system functionality based on actors and their goals to sequence diagrams that capture the communication between objects through sequences of messages.

We focus on UML Activity Diagrams (UML ADs), which are primarily used to model operational business processes. An Activity Diagram is a graph that comprises activities to depict the essential steps of processing. Atomic activities are referred to as actions. Activities may be assigned to roles via a swimlane concept and objects are used to represent the processed data. Sequencing of activities is encoded by control flows, decision and merge nodes model exclusive disjunctive splitting and merging behaviour, fork and join nodes implement conjunctive behaviour. There are initial nodes and final nodes to depict the initialisation and completion of a business process. Exception handling is encoded by interruptible regions and interrupting edges.

As for BPMN collaboration diagrams, execution semantics of UML ADs are defined informally and assume a notion of token flow. In contrast to BPMN collaboration diagrams and EPCs, UML ADs emphasise the coupling of control flow and data flow aspects. Besides control flow, object flow is modelled between activities, which incorporates objects to represent the data, pins to model input and output data dependencies of activities, and data streaming mechanisms. Further, the execution semantics distinguishes control token and data tokens. Both token types are handled differently by certain constructs. For instance, a join node that merges parallel branches forwards all data tokens separately, but synchronises control tokens.

Execution semantics of UML ADs have been (partially) formalised using abstract state machines [58]. Other work [146] relies on the Statemate semantics of statecharts [198]. Further, there exists a partial formalisation using coloured Petri nets [431]. Still, there is a notable gap between various high-level constructs of UML ADs and Petri net concepts [433].

An introduction to the concepts of UML and their application to business modelling, including but not restricted to Activity Diagrams, can be found in [143]. We illustrate the notations of UML ADs with Figure 7. Actions are denoted by rounded rectangles. The initial node is depicted by a filled circle, the final node is captured by a filled circle framed by another circle. Decision and merge nodes are visualised by diamonds, fork and join nodes by filled bars. Again, the model depicts a lead-to-

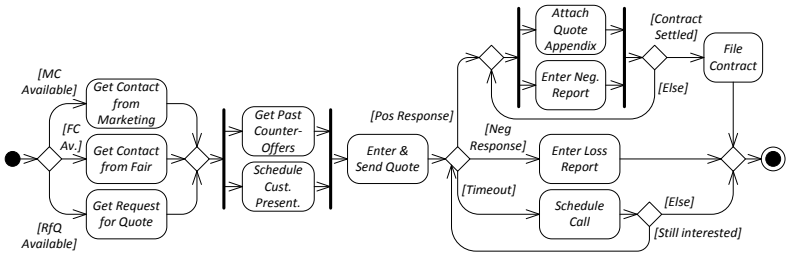


Figure 7: A model of a lead-to-order process depicted as a UML Activity Diagram.

order process. It is similar, but not semantically identical to those captured earlier in BPMN and as an EPC.

The relation between BPMN process models in terms of collaboration diagrams and UML Activity Diagrams has been investigated in [86]. This work also introduces a basic model-to-model transformation from BPMN to UML ADs.

Web Service Business Process Execution Language

The Web Service Business Process Execution Language [17], or BPEL for short, has been standardised by the OASIS consortium[5]. BPEL is an XML based language for the description of business processes that is grounded on web service interactions. Any functionality is imported and exported solely through web service interfaces. Since BPEL models coordinate service interactions, BPEL is often referred to as an orchestration language.

There are two kinds of BPEL process models. Non-executable *abstract* BPEL processes focus on the business protocol. They provide a partial specification and hide operational details. Abstract BPEL processes are typically applied as role-models for process implementations and serve as contracts between different participants in a business interaction. Besides abstract processes, there are fully-specified *executable* processes. Those capture the behaviour of a particular participant of a business interaction. Executable BPEL processes are intended to be deployed and executed on a workflow engine. As a consequence, executable BPEL processes contain all operational details, from a web service binding and message correlation mechanism over a control flow specification to the data access handling. In the remainder, we focus on the model elements to access functionality and to capture the control flow.

Atomic functionality in BPEL processes is provided by basic activities. Those enable to, synchronously or asynchronously, call a web service (invoke), provide a web service (receive and reply), update internal data (assign), trigger internal exceptions

5 http://www.oasis-open.org/

(throw and rethrow), or pause or terminate the execution (wait and exit). The notion of a scope provides a concept for structuring a BPEL process hierarchically and to control data access along with exception, event, and termination handling. Further, BPEL supports two conceptually different paradigms to define control flow dependencies between basic activities. Historically, these paradigms stem from the Web Services Flow Language (WSFL) [269] and XLANG [440], which are two successors of BPEL for the description of web service compositions. Following on XLANG, BPEL allows for the well-structured specification of control flow by nesting structured activities. Those are typed to represent sequences (sequence), conditional execution (if-then-else), repetitive behaviour (while and repeat until), selective event processing (pick), concurrent execution (flow), or branch processing (for each). On the other hand, BPEL supports the definition of (acyclic) graph-structured control flow dependencies by control links between activities within a flow activity. Hence, BPEL is not block-structured in the strict sense, but inherits graph-structured capabilities from WSFL.

The BPEL specification provides an informal description of execution semantics. Formalisations of BPEL have been presented in numerous papers, see [445] for an overview. BPEL formalisations have been grounded on abstract state machines [155, 158], Petri nets [341, 211, 281], and process algebra [161, 496, 288]. Only a few formalisations, e. g., [281], aim at capturing the complete spectrum of BPEL constructs.

BPEL has an XML-based syntax and does not standardise any visual representation of the model elements. For illustration purposes, we depict a simplified BPEL process in Figure 8. This model shows only the skeleton of a process and does not contain all details required for execution on a workflow engine. Using the example of the lead-to-order process, it illustrates the basic concepts to implement control flow dependencies in BPEL. We use a pick activity to instantiate the process upon reception of a message. Either a contact from marketing, a contact from a fair, or a request for quote is expected. Note that we duplicated the concurrent execution of the activities to check past offers and to schedule a customer presentation. This keeps the control flow structure clearly arranged. Then, the process continues by reacting to the customer's choice or the time-out accordingly. Our example relies on the block-structured concepts that BPEL inherits from XLANG. However, large parts of the process could also be modelled following the graph-structured paradigm, by using control links between activities.

In relation to the aforementioned languages, BPEL is often perceived as the technical back-end. The definition of BPMN in particular was motivated by the absence of any notation for

```
<?xml version='1.0' encoding='UTF-8'?>
<bp:process exitOnStandardFault='yes' name='LeadToOrder'
  suppressJoinFailure='yes' targetNamespace='Example'
  xmlns:bp='http://docs.oasis-open.org/wsbpel/2.0/process/abstract'
       xmlns:tns='Example'>
  <bp:sequence>
    <bp:pick createInstance='yes'>
      <bp:onMessage name='MarketingContact'>
        <bp:sequence>
          <bp:flow>
            <bp:invoke name='GetPastCounterOffers'/>
            <bp:invoke name='ScheduleCustomerPresentation'/>
          </bp:flow>
        </bp:sequence>
      </bp:onMessage>
      <bp:onMessage name='FairContact'>
        <bp:sequence>
          <bp:flow>
            <bp:invoke name='GetPastCounterOffers'/>
            <bp:invoke name='ScheduleCustomerPresentation'/>
          </bp:flow>
        </bp:sequence>
      </bp:onMessage>
      <bp:onMessage name='RequestForQuote'>
        <bp:empty />
      </bp:onMessage>
    </bp:pick>
    <bp:invoke name='EnterQuoteDetails'/>
    <bp:invoke name='SendQuote'/>
    <bp:repeatUntil>
      <bp:pick>
        <bp:onMessage name='PositiveResponse'>
          <bp:sequence>
            <bp:while>
              <bp:flow>
                <bp:invoke name='EnterNegotationReport'/>
                <bp:invoke name='AttachQuoteAppendix'/>
              </bp:flow>
            </bp:while>
            <bp:invoke name='FileContract'/>
          </bp:sequence>
        </bp:onMessage>
        <bp:onMessage name='NegativeResponse'>
          <bp:invoke name='EnterLossReport'/>
        </bp:onMessage>
        <bp:onAlarm name='TwoWeeks'>
          <bp:scope>
            <bp:invoke name='ScheduleCall'/>
          </bp:scope>
        </bp:onAlarm>
      </bp:pick>
    </bp:repeatUntil>
  </bp:sequence>
</bp:process>
```

Figure 8: A lead-to-order process model in BPEL syntax.

BPEL process models, cf., [514]. Hence, the BPMN specification sketched a transformation from BPMN to BPEL already in its first version. The definition of a full-fledged transformation from BPMN to BPEL, or even complete round-tripping, turned out to be challenging and has been addressed in various papers [373, 340, 339, 166, 497, 245]. With BPMN in its version 2.0, the scope and applicability of such a transformation has been clarified. Transformations between EPCs and BPEL processes [244, 310], and between UML ADs and BPEL processes [202, 240, 534, 482] have also been discussed extensively.

2.2 NET SYSTEMS

The process description languages discussed in the previous section are widely used in industry. For an analysis of behaviour consistency of related process models, however, these languages are not well-suited. They are missing formal semantics and techniques for their analysis. Therefore, we focus on a well-established formalism for the description of process models, i. e., Petri net systems. This section is dedicated to the formal definition of net systems. First, we recall basic mathematical notions and their notations. Then, we define the syntax and semantics of net systems. Finally, we elaborate on structural and behavioural classes of net systems.

Preliminaries

For the discussion of net systems, we need basic mathematical notions. We shortly recall the notions and notations used in the remainder of this thesis.

For a set S, we refer to its cardinality as $|S|$. The power set of S is denoted by $\wp(S)$. Given two sets, equivalence is denoted by $=$, inclusion is denoted by $\subseteq$, and proper inclusion by $\subset$. Further, $\cap$ creates the intersection of two sets, $\cup$ creates the union of two sets, and $\times$ creates the Cartesian product of a set.

The set of natural numbers, excluding 0, is denoted by $\mathbb{N}$. The natural numbers, including 0, are denoted by $\mathbb{N}_0$. We use the following notations for Boolean algebra. The conjunction of Boolean statements is denoted by $\wedge$, the disjunction by $\vee$. By $\Rightarrow$, we refer to an implication between Boolean statements, $\Leftrightarrow$ denotes the equivalence of Boolean statements.

An n-ary relation $R \subseteq (S_1 \times S_2 \times \ldots \times S_n)$, $n \in \mathbb{N}$, is a set of n-tuples, such that the k-th component, $k \in \mathbb{N}$, $k \leqslant n$, of an n-tuple is taken from S_k. For $n = 2$, R is called binary and is a subset of the Cartesian product over which it is defined. For $(x, y) \in R$, we also write $x \; R \; y$. For $(x, y) \notin R$, in turn, we also write $x \not R \; y$. The identity relation id_S over a set S is a binary

relation, defined as $id_S = \{(x,x) \mid x \in S\}$. For a relation $R \subseteq (S_1 \times S_2)$, the inverse relation R^{-1} is defined as $R^{-1} = \{(x,y) \in (S_2 \times S_1) \mid y\ R\ x\}$.

A binary relation $R \subseteq (S \times S)$ over a set S is symmetric, if $\forall\, x,y \in S$ it holds $(x\ R\ y) \Rightarrow (y\ R\ x)$. The relation R is asymmetric, if it is not symmetric. It is antisymmetric, if $(x\ R\ y \wedge y\ R\ x)$ implies $x = y$. The relation R is reflexive, if $\forall\, x \in S$ it holds $x\ R\ x$. The relation R is irreflexive, if it is not reflexive. The relation R is transitive, if $\forall\, x,y,z \in S$ it holds $(x\ R\ y \wedge y\ R\ z) \Rightarrow (x\ R\ z)$. With R^+, we denote the irreflexive transitive closure of relation R and with R^* we refer to the reflexive transitive closure.

A binary relation $R \subseteq (S_1 \times S_2)$ is total from S_1 to S_2, if $\forall\, x \in S_1\ [\ \exists\, y \in S_2\ [\ x\ R\ y\ ]]$. The relation R is surjective, if it is total from S_2 to S_1. The relation R is functional, if $\forall\, a \in S_1, x, y \in S_2\ [\ (a\ R\ x) \wedge (a\ R\ y) \Rightarrow (x = y)\ ]$. The relation R is injective, if $\forall\, a,b \in S_1, x \in S_2\ [\ (a\ R\ x \wedge b\ R\ x) \Rightarrow (a = b)\ ]$. The relation R is bijective, if it is total in either direction, functional, and injective.

A binary relation $f \subseteq (S_1 \times S_2)$ is a function from S_1 to S_2, denoted by $f : S_1 \mapsto S_2$, if it is total from S_1 to S_2 and functional. For the function f, S_1 is the domain and S_2 is the codomain. For $(x,y) \in f$ with f being a function, we also write $f(x) = y$.

A binary relation $R \subseteq (S_1 \times S_2)$ may be identified by its characteristic function $f_R : (S_1 \times S_2) \mapsto \{0,1\}$.

A function $f : \{1,\ldots,n\} \mapsto S$ is a finite sequence over a set S. For a sequence $f : \{1,\ldots,n\} \mapsto S$, we also write $f = \langle s_1, s_2, \ldots, s_n \rangle$ with $f(i) = s_i$, $i \in \mathbb{N}$, and $1 \leqslant i \leqslant n$. The length of the sequence f is n. For a finite sequence $f = \langle s_1, s_2, \ldots, s_n \rangle$, $g = \langle s_j, s_{j+1}, \ldots, s_k \rangle$, $j,k \in \mathbb{N}$, $1 \leqslant j \leqslant k \leqslant n$, is a subsequence.

Net Syntax

Petri nets are grounded on the ideas Carl Adam Petri presented in his seminal doctoral thesis [353] in 1962. Aiming at asynchronous communication, he proposed a behavioural formalism that knows only local actions that have local causes. Since then, this theory of nets has seen a huge uptake in Computer Science. There is a large body of results and analysis techniques for Petri nets, see [329]. Further, they have been applied in such diverse areas as hardware design [93] and bioinformatics [347, 528]. Exhaustive introductions to Petri nets can be found in [380, 382, 119, 381].

A Petri net, or *net* for short, is a bipartite directed graph consisting of places and transitions. The directed edges, called flows, connect places with transitions and transitions with places.

Definition 2.2.1 (Net)
A *net* is a tuple $N = (P, T, F)$ with P and T as finite disjoint sets

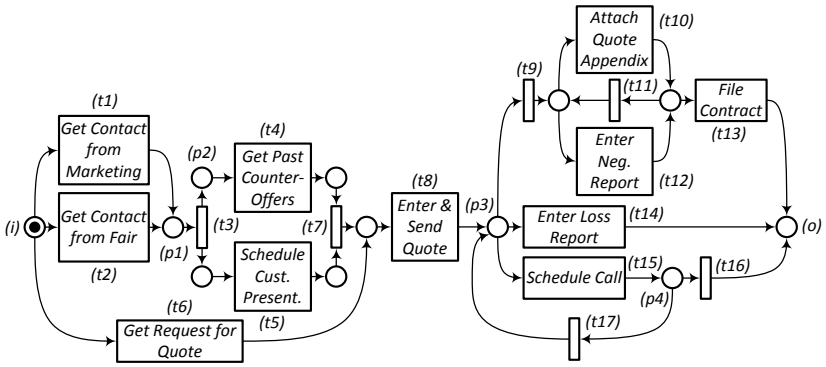


Figure 9: A model of a lead-to-order process depicted as a marked net, a net system.

of places and transitions, and $F \subseteq (P \times T) \cup (T \times P)$ as the flow relation.

We illustrate the visualisation of nets with the example given in Figure 9. This example takes up the lead-to-order process that was used to exemplify process description languages in the previous section. Figure 9 shows a marked net. Neglecting the marking at this stage, the net comprises places, denoted by circles, and transitions, denoted by rectangles, that are connected by directed edges representing flows.

In our example, several transitions carry labels, e. g., 'Get Contact from Fair', whereas others are not labelled at all and are depicted by smaller rectangles. Throughout this thesis, we use this convention to indicate which transitions carry a meaning according to the domain of the model. Transitions that are present in the model only for syntactical reasons are depicted without a label and in smaller size. However, this is only a convention for visualisation. Unless explicitly stated otherwise, we assume nets with unique transitions as our formal model.

For a net $N = (P, T, F)$, we write $X = (P \cup T)$ for all nodes. As introduced earlier, we use F^+ to denote the irreflexive transitive closure of F. We define further basic syntactical concepts of nets as follows.

Definition 2.2.2 (Net Syntax)
Let $N = (P, T, F)$ be a net and $X = (P \cup T)$ all nodes.

- For a node $x \in X$, $\bullet x = \{y \in X \mid (y, x) \in F\}$ is the *pre-set* of x and $x\bullet = \{y \in X \mid (x, y) \in F\}$ is the *post-set* of x.
- A tuple $N' = (P', T', F')$ is a *subnet* of a net $N = (P, T, F)$, iff $P' \subseteq P$, $T' \subseteq T$, and $F' = F \cap ((P' \times T') \cup (T' \times P'))$. N' is a *partial* subnet of N, iff $F' \subseteq F \cap ((P' \times T') \cup (T' \times P'))$.
- A *path* of length $n \in \mathbb{N}$, it holds $n > 1$, is a sequence $\pi_N = \langle x_1, \ldots, x_n \rangle$, which satisfies $(x_1, x_2), \ldots, (x_{n-1}, x_n) \in F$. For a path $\pi_N = \langle x_1, \ldots, x_n \rangle$, we also write $\pi_N(x_1, x_n)$. We write

$\pi_N\{x_1, x_n\} = \{x_1, \ldots, x_n\}$ for all nodes that are part of the path $\pi_N(x_1, x_n)$.
- A *subpath* π'_N of a path π_N is a subsequence that is itself a path.
- A path $\pi_N(x_1, x_n)$ is a *circuit*, iff $(x_n, x_1) \in F$ and no node occurs more than once in the path.

For the net in Figure 9, we annotated several places and transitions with numbers for illustration purposes. For transition t1 the post-set contains a single place, p1. The pre-set of place p1 comprises two transitions, t1 and t2. The sequence of nodes $\pi_1 = \langle t1, p1, t3, p2 \rangle$ is a path, the sequence $\pi_2 = \langle t1, p1 \rangle$ is a subpath of this path, the sequence $\langle t1, p1, p2 \rangle$ is not a subpath of π_1. The path $\pi_3 = \langle p3, t15, p4, t17 \rangle$ is a circuit.

Net Semantics

Having defined the syntax of a net, we turn to its semantics. Semantics of nets are defined as a token game. A state of a net is described by a set of tokens that are distributed among its places. Such a distribution of tokens is called marking of a net. State transitions are represented by changes of the marking of a net. A transition is enabled in a certain marking if the places of its pre-set carry at least one token. Then, the transition can be fired. Firing of a transition means consuming tokens from places in the pre-set and producing tokens on places in the post-set of the transition. As such, firing of a transition in a marking leads to a new marking, i. e., it constitutes a state transition. Formally, we define semantics as follows.

Definition 2.2.3 (Net Semantics)
Let $N = (P, T, F)$ be a net.
- $M : P \mapsto \mathbb{N}_0$ is a *marking* of N, $\mathbb{M}$ denotes all markings of N. $M(p)$ returns the number of *tokens* in place p.
- For a place $p \in P$, M_p denotes the marking when place p contains one token and all other places contain no tokens. For a transition $t \in T$, M_p denotes the marking when all places $p \in {\bullet}t$ contain one token and all other places contain no tokens.
- For any two markings $M, M' \in \mathbb{M}$, we write $M < M'$, iff $M(p) < M'(p)$ for all places $p \in P$. We write $M \leqslant M'$, iff $M(p) \leqslant M'(p)$ for all places $p \in P$.
- For any transition $t \in T$ and any marking $M \in \mathbb{M}$, t is *enabled* in M, denoted by $(N, M)[t\rangle$, iff $\forall\, p \in {\bullet}t\ [\, M(p) \geqslant 1\,]$.
- If $t \in T$ is enabled in M, then it can *fire*. Firing of t, denoted by $(N, M)[t\rangle(N, M')$, leads to a new marking M'. This marking M' is defined by $M'(p) = M(p) - f_F(p, t) + f_F(t, p)$, with f_F as the characteristic function of F, for each place $p \in P$.

- A sequence of transitions $\sigma = \langle t_1 \dots t_n \rangle$, $n \in \mathbb{N}_0$, is a *firing sequence*, iff there exist markings $M_0, \dots, M_n \in \mathbb{M}$, such that for all $1 \leqslant i \leqslant n$ it holds $(N, M_{i-1})[t_i\rangle(N, M_i)$. We say that σ is enabled in M_0, denoted by $(N, M_0)[\sigma\rangle$. For $n = 0$, we refer to $\sigma = \langle\rangle$ as the empty firing sequence.
- For any two markings $M, M' \in \mathbb{M}$, M' is *reachable* from M in N, denoted by $M' \in [N, M\rangle$, iff there exists a firing sequence σ leading from M to M'.
- A *net system*, or a *system*, is a pair (N, M_0), where N is a net and M_0 is the *initial marking* of N.

A marking of a net system is visualised by depicting each token with a black dot inside the circle of the respective place. In Figure 9, only the place i carries a token.

As stated before, semantics of a net system describe states and state transitions. Hence, a net system describes a *state space* that may also be captured by a labelled transition system (LTS). Labelled transition systems are one of the most general behavioural models. An LTS is a directed graph that comprises a set of states, a set of actions, a transition relation that associates an action to a source state and a target state, and an initial state. The LTS for a net system contains a state for each reachable marking. The initial marking of the system is the initial state of the LTS. Each firing of an enabled transition in some marking in the net system is represented by a transition in the LTS – the action references the respective transition in the system and the source and target states correspond to the markings before and after firing of the transition, respectively.

Definition 2.2.4 (State Space of a Net System)
A labelled transition system is a tuple $TS = (\mathcal{S}, s_0, \Lambda, \mathcal{F})$, such that $\mathcal{S}$ is a set of states, $s_0 \in \mathcal{S}$ is an initial state, Λ is a set of labels, $\mathcal{F} \subseteq \mathcal{S} \times \Lambda \times \mathcal{S}$ is a labelled transition relation.
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$. The *state space* of S is a transition system $TS_S = (\mathcal{S}, s_0, \Lambda, \mathcal{F})$ derived as follows.

- Each marking reachable from the initial marking is taken as a state, $\mathcal{S} = [N, M_0\rangle$, and the initial marking is taken as the initial state, $s_0 = M_0$.
- Each transition is taken as a label, $\Lambda = T$.
- Each transition that can be fired in a certain marking is taken as a labelled state transition, $(M, t, M') \in \mathcal{F}$, iff there exists a transition $t \in T$ and $(N, M)[t\rangle(N, M')$.

The state space of a net system is also known as the *reachability graph* of the net system. Figure 10 exemplifies the state space for the example net system given in Figure 9. Here, circles denote states and labelled edges represent state transitions with the ac-

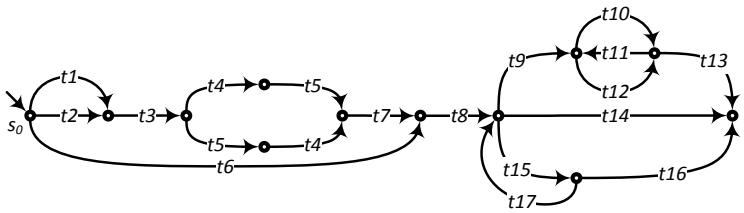


Figure 10: The state space of the net system depicted in Figure 9.

cording action. The initial state is highlighted with a small edge pointing to state s_0.

Besides state-based semantics, net systems may be interpreted according to *trace semantics* [302, 121]. Then, the behaviour is defined in terms of all firing sequences of a net system that start in the initial marking. Such a firing sequence is referred to as a *trace* of the net system. In other words, a trace is a path of the state space of the net system.

Definition 2.2.5 (Traces of a Net System)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$. The set of *traces* $\mathcal{T}$ of S contains all firing sequences σ, such that $(N, M_0)[\sigma\rangle$.

The set of all traces is also called the system's *language*. It corresponds to all paths of the state space of the net system. A state space may comprise an infinite number of states. Even a finite number of states, however, may result in an infinite set of traces. The latter is illustrated by our example in Figure 9 and Figure 10. Due to the circuit in the net system, there are infinitely many traces. Examples for traces are, e. g., $\sigma_1 = \langle t1\rangle$, $\sigma_2 = \langle t1, t3, t4, t7, t9\rangle$, and $\sigma_3 = \langle t1, t3, t4, t7, t9, t10, t11, t10, t11, t10\rangle$.

Finally, we clarify the terminology with respect to net systems that model a business process. We assume the following interpretation. The net system is a process model, transitions are either business activities or carry no semantics in terms of the modelled business process, and a marking of a net corresponds to a state of an instance of the process model. For a firing sequence of transitions of a net system, we speak of an execution sequence of activities of a process model. We will use the Petri net terminology and the process model terminology interchangeably in the remainder of this thesis. We discuss formal results in Petri net terminology. At certain stages, however, we stick to the terms process model and activities. That is to highlight that our arguments should be understood in the context of using net systems to model business processes.

Structural Classes of Net Systems

In this thesis, we use net systems that show certain structural properties. These properties define structural classes of net sys-

tems. Several of our results are obtained for free-choice nets. These nets show a separation of synchronisations and conflicts, which led to a large theory on the analysis of their behavioural properties. As such, they are often quoted as a good compromise between expressiveness and the ability to do efficient behavioural analysis. Further, we consider two subclasses of free-choice nets, i. e., T-nets and S-nets. Introductions to free-choiceness and details on their properties can be found in [441, 49, 119].

Definition 2.2.6 (T-Net, S-Net, Free-Choice Net)
Let $N = (P, T, F)$ be a net.

- The net N is a *T-net*, iff $\forall\, p \in P\ [\, |\bullet p| \leqslant 1 \geqslant |p \bullet|\,]$.
- The net N is an *S-net*, iff $\forall\, t \in T\ [\, |\bullet t| = 1 = |t \bullet|\,]$.
- The net N is *free-choice*, iff $\forall\, p \in P, t \in T$, $(p, t) \in F$ implies $\bullet t \times p\bullet \subseteq F$.

A net system $S = (N, M_0)$ is a free-choice, S-, T-system, iff N is a free-choice, S-, T-net. Note that our definition of free-choiceness is commonly used [119], but historically referred to as *extended free-choiceness* [49]. This stems from a definition of free-choiceness that is more restrictive than the presented one. Still, in terms of the ability to do efficient behavioural analysis, both definition achieve the same effect. Any extended free-choice net can be transformed into a behaviour equivalent free-choice net [49] – here, behaviour equivalence assumes that transitions inserted by this transformation are ignored. Hence, the term free-choiceness is often used to refer to the presented definition.

We also rely on the notion of a workflow (WF-) net [447]. This class has been proposed explicitly for modelling and analysing business processes. WF-nets require the existence of a dedicated initial place and a dedicated final place. Those represent initialisation and completion of the business process. The definition of WF-nets also requires all nodes to be on a path from the initial place to the final place. This requirement translates into strong connectivity of the short-circuit net. The latter is obtained by connecting the final place with the initial place through a fresh transition.

Definition 2.2.7 (WF-Net)
A net $N = (P, T, F)$ is a *workflow (WF-) net*, iff N has an *initial* place $i \in P$ with $\bullet i = \emptyset$, N has a *final* place $o \in P$ with $o\bullet = \emptyset$, and the *short-circuit* net $N' = (P, T \cup \{t^*\}, F \cup \{(o, t^*), (t^*, i)\})$ of N is strongly connected.

The initial and the final place of a WF-net characterise an initial and a final marking. A *WF-system* is a net system $S = (N, M_i)$ with N being a WF-net and i being its initial place. The marking M_o of S is referred to as the final marking of S. For a WF-system $S = (N, M_i)$ and N' as the short-circuit net of N, the system $S' =$

(N', M_i) is the short-circuit system. The structural properties are directly lifted to WF-systems. That is, a WF-system $S = (N, M_i)$ is free-choice, a workflow S-system, or a workflow T-system, iff N is free-choice, an S-net, or a T-net.

The net system shown in Figure 9 is a free-choice WF-system. We annotated the initial place with i and the final place with o.

Behavioural Properties of Net Systems

After we discussed structural classes of net systems, we recall elementary behavioural properties that are of relevance for our work. First, boundedness restricts the behaviour of a net system to a set of finite markings. That is, the state space of the net system is finite, i. e., the respective LTS has a finite number of states. A different characterisation of boundedness assumes that there is an upper bound for the number of tokens for any place in all markings reachable from the initial marking. Safeness sets this bound to one, i. e., no place is marked with more than one token.

Definition 2.2.8 (Boundedness and Safeness)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$.

- The system S is *bounded*, iff the set $[N, M_0\rangle$ is finite, and *unbounded* otherwise.
- The system S is *safe*, iff for all markings $M \in [N, M_0\rangle$ and places $p \in P$ it holds $M(p) \leqslant 1$, and *unsafe* otherwise.

The net system depicted in Figure 9 is bounded and safe. Further, we recall common properties related to liveness. A transition is dead in a marking, if it cannot be fired as part of any firing sequence starting in this marking. A marking is dead, if it does not enable any transition. A transition is live, if it may be enabled from every marking reachable from the initial marking. A net system is live, if every transition may be fired again, i. e., every transition is live.

Definition 2.2.9 (Liveness Properties)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$.

- A transition $t \in T$ is *dead* in a marking M of S, iff for all markings $M' \in [N, M\rangle$ it holds that t is not enabled in M'.
- A marking M of S is *dead*, iff it does not enable any transition $t \in T$.
- A transition $t \in T$ is *live*, iff for all markings $M \in [N, M_0\rangle$ there is a marking $M' \in [N, M\rangle$ such that $(N, M)[t\rangle$.
- The system S is *live*, iff every transition $t \in T$ is live.

Transition t4 of the net system depicted in Figure 9 is not dead in the initial marking. Transition t4 is dead in the marking that marks only place p3, reached, e. g., by the firing sequence $\sigma =$

$\langle t6, t8 \rangle$. The marking M_o that marks only the place o is dead. No live net system shows a dead marking [119], so that our example net system is not live.

Liveness and boundedness properties are particularly interesting for the class of WF-systems, for which there exists the soundness property. The latter requires WF-systems (1) to always terminate, and (2) to have no transitions that are dead in the initial marking [448]. Both requirements imply proper termination of the WF-systems, i. e., for all reachable markings holds that a token in the final place implies the absence of tokens for all other places.

Definition 2.2.10 (Soundness)
A WF-system (N, M_i) with the initial place i and the final place o is *sound*, iff

- for all markings $M \in [N, M_i\rangle$ there is a firing sequence σ such that $(N, M)[\sigma\rangle(N, M_o)$, and
- for all transitions $t \in T$, there exists a marking $M \in [N, M_i\rangle$ and $(N, M)[t\rangle$.

The soundness property is closely related to the liveness and boundedness properties introduced earlier. A system is *sound*, if and only if the corresponding short-circuit system is live and bounded [448, 467].

2.3 FROM PROCESS DESCRIPTIONS TO NET SYSTEMS

Net-based formalisations are available for all process description languages mentioned earlier, for BPMN [125], EPCs [116, 472], UML ADs [431], and BPEL [341, 211, 281]. Note that these formalisations of execution semantics differ with respect to their coverage of the languages. An overview of net-based formalisations of process description languages is provided in [283].

In the previous section, we introduced several structural and behavioural properties that classify net systems. Figure 11 takes up the properties that are most relevant for our work, in particular bounded net systems and sound free-choice WF-systems. The properties are mostly orthogonal, apart from the fact that soundness has been defined for WF-systems and boundedness is a prerequisite for soundness. In the remainder of this section, we discuss net-based formalisations of process description languages against the dimensions illustrated in Figure 11. We restrict the discussion to the control flow aspects. For details on net-based formalisations of object life-cycles and data access semantics we refer the reader to [451, 26]. Also, there are many types of high-level Petri nets that consider data aspects, such as coloured Petri nets (CPNs) [220], workflow nets with data

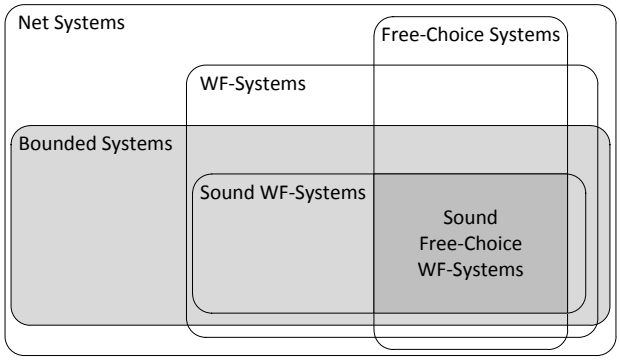


Figure 11: Classification of net systems by structural and behavioural properties.

(WFD-nets) [419], or dual workflow nets (DFNs) [157]. To keep the focus on the behavioural aspects, we use net systems as introduced in the previous section.

Net Systems. Earlier, we clarified that transitions of net systems are intended to model activities of a process model. In principle, all constructs of a process model for which the behaviour can be split up into atomic steps can be represented in this way. For instance, events in BPMN or EPCs, the initial and final nodes in UML ADs, or data assignments in BPEL can also be represented by a net transition. If needed, even a more fine-grained formalisation is possible. Instead of representing an activity with a single transition, all transitions of its life-cycle may be modelled by separate transitions. The latter approach has been realised in the Yet Another Workflow Language (YAWL) [439], a process description language that is inspired by the net system formalism.

The opposite case, abstracting a semantically rich construct of a process description language by one or more transitions in the net system, may also be required. High-level concepts, e. g., multi-instance activities configured towards a certain use case or exception handling constructs, allow for expressing complex behavioural dependencies in a process model. In many cases, those cannot be represented in net systems directly. Then, one solution is to approximate the concepts with a coarse-grained formalisation, in which the respective behaviour is only partially covered in the net system. For BPMN, EPCs, and BPEL several formalisation challenges are discussed in [283].

WF-Systems. The workflow net structure imposes two requirements. First, there has to be a dedicated entry and a dedicated exit that represent the initialisation and completion of a process, i. e., an initial and a final place in the respective net. Some process description languages explicitly enforce such a structure, e. g., UML Activity Diagrams. Others, such as BPMN and EPCs, support the definition of multiple start events and end events. In these cases, restructuring techniques [480] may be applied to

normalise the model structure before transforming it to a net system. In certain cases, however, this may not be possible. On the one hand, we mentioned that instantiation semantics may not be well defined if there are multiple (EPC) start events [109]. A solution to this issue that selects sound instantiation scenarios can be found in [361]. On the other hand, constructs that spawn execution if the process is in a certain region of states or non-local termination semantics can hardly be transformed to WF-systems. The non-interruptive event handling mechanism of BPMN is an example for the former. Also, a BPMN termination end event leads to immediate termination of the process instance, independent of the state of the instance. Encoding such behaviour using net systems is challenging and may not be possible in all cases.

Second, the workflow net structure requires that all nodes are on a path from the entry to the exit of the process model. Once potential issues related to the existence of such an entry and an exit are resolved, this requirement is rarely an issue. Albeit not enforced by all syntax definitions, it can be seen as common practise to define a process model as a connected graph that is even strongly connected when the exit is connected to the entry.

Free-Choice-Systems. Whether the net-based formalisation of a process model yields a free-choice net mainly depends on the set of used control flow routing elements. Means to encode disjunctive (XOR) and conjunctive (AND) splitting or joining semantics in process models are transformed to free-choice net constructs in a straight-forward manner. Concepts to define inclusive disjunctive (OR) behaviour are often not transformed due to their non-local semantics. If transformed, the resulting net is typically non-free-choice since it represents all combinations of possible activations, see [472]. Further, advanced control flow routing needed to express exception handling and event processing often results in non-free-choice nets [283]. The formalisation of interrupting events for subprocesses in BPMN proposed in [125] or the formalisation of fault, compensation, and termination handlers in BPEL presented in [281] are examples for transformations that result in non-free-choiceness.

Boundedness & Soundness. The behavioural properties of a net system follow directly from the process model that was transformed into the net system. Since the net system should represent the same behaviour of the process model once data-related aspects are neglected, unboundedness of the process model implies unboundedness of the net system. Unboundedness of a process model is not necessarily a behavioural anomaly that should be avoided. Event-based mechanisms that allow for non-interruptive instantiation of concurrent execution, e. g., non-interruptive event handlers in BPMN and BPEL, may lead to an

infinite state space of a process model. Such cases may require abstracting from the possibility of infinitely many instantiations to come to a bounded net system. Further, the definition of a transformation for constructs that realise inclusive disjunctive (OR) semantics is important in this regard. For instance, different net-based formalisations for the OR-connector of EPCs have been presented, which create a potentially unbounded system [116] or ensure safeness of the created net system [472].

Once a process model meets the requirements needed to transform it into a WF-system, the soundness criterion can be lifted to the process model. Having discussed boundedness already, we turn the focus to properties related to liveness, i. e., there has to be an option to complete the process from any reachable state and every activity may be enabled in some reachable state. In principle, these properties should be consistently satisfied or violated in the process model and the net system derived as its formalisation once data-related aspects are neglected. For certain constructs, however, the operationalisation of the transformation impacts on the soundness property. Again, the way inclusive disjunctive (OR) semantics is encoded is an example for this issue. The formalisation of the OR-connector of EPCs presented in [116] may result in a net system that shows a deadlock, a dead marking that is not the final marking. Since this violates the soundness property, an adapted correctness criterion, relaxed soundness, has been proposed [116]. However, under certain assumptions, if OR-splitting and OR-merging constructs are well-structured, a different formalisation may be applied. Then, inclusive disjunctive constructs of a process model may be transformed such that the resulting net system is sound.

2.4 DISCUSSION

In this chapter, we discussed the subjects of our work, process models. We reviewed four process description languages often used in practise to highlight their commonalities and differences. To this end, our focus has been on the concepts to capture the control flow between activities. We argued that models defined in these languages are not suited for behavioural analysis. They are missing formal execution semantics and established techniques for their analysis. Therefore, we introduced the Petri net formalism as the grounding of our work. Although net systems may directly be used to model business processes – this has been the motivation for the definition of workflow systems, a subclass of generic net systems – they have seen little uptake in industry for this purpose. Hence, models captured in languages such as BPMN or EPCs have to be transformed into net systems before analysis.

The essential control flow routing elements of the discussed languages can be transformed into net system concepts in a straight-forward manner. For some languages, there are even formalisations that claim feature-completeness and consider advanced concepts. For instance, non-interruptive event-handling as defined by BPEL has been formalised in [281]. We observe a conceptual closeness of the presented process description languages and the net system formalism. They follow the same procedural modelling paradigm. Further, the definition of semantics is often inspired by the token flow concept known from Petri nets. The conceptual closeness is also witnessed by the Yet Another Workflow Language (YAWL) [439]. It emerged as a process description language that is grounded on net systems and extends them with features that are specific to business processes, an OR-join to realise inclusive disjunctive merging of behaviour, cancellation regions to implement exception handling, and dedicated multiple instance activities. Note that we discussed exactly these concepts as being particularly challenging for net-based formalisations. However, depending on the purpose of process modelling, a large share of process models observed in practise is created using solely elementary control flow routing elements [538].

Besides the conceptual closeness, we chose net systems as our formal basis since there is a large theory on their analysis along with tool support. The Petri Nets World[6] lists more than 8500 publications related to net systems along with a large number of modelling and analysis tools. As such, the net system formalism is unique. Other formalisms have been proposed to model business processes or to formalise process models. Those may have certain advantages for dedicated analysis questions, but fall short of a comprehensive framework as it exists for net systems. As an example, consider the π-calculus [324, 323, 404], a process algebra that gained remarkable attention in the field of BPM around 2005 [367, 496, 288, 368, 331], also referred to as the π-*hype* [468]. The name passing feature of the π-calculus allows for modelling dynamic service binding in an intuitive way. Representing dynamic service binding with net systems is much more challenging and typically realised by extending the formalism [111]. However, many of the aforementioned formalisation issues are not solved in the π-calculus either. For instance, the formalisation of inclusive disjunctive (OR) merging of behaviour presented in [367] outsources the decision of which branches to merge to a run time executing the model. Also, there are only a few tools available that allow for analysing π-calculus models, e. g., the Mobility Workbench[7] and the Advanced Bisimulation

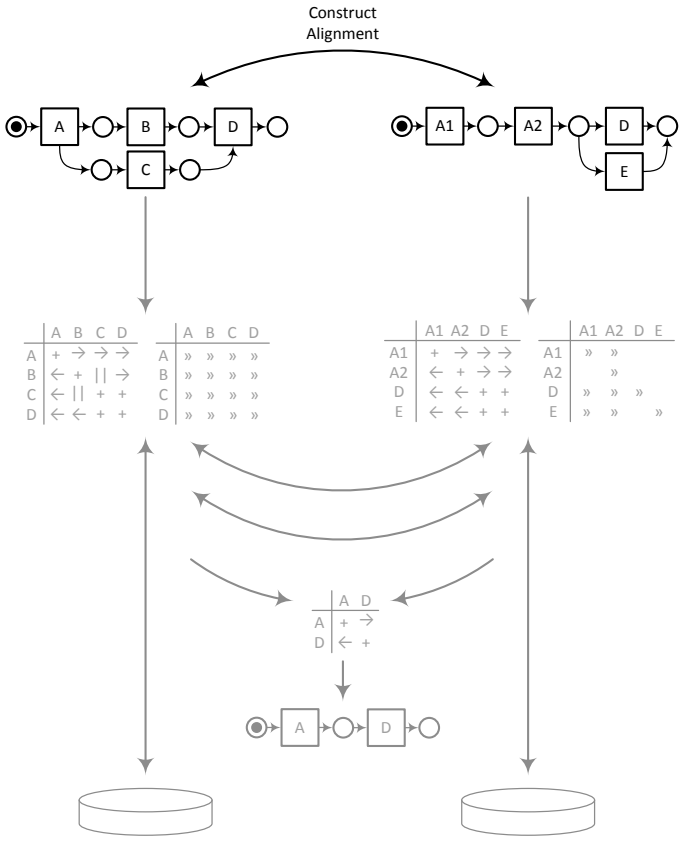
6 http://www.informatik.uni-hamburg.de/TGI/PetriNets/
7 http://www.it.uu.se/research/group/mobility/mwb/

Checker[8]. Therefore, we decided for net systems as the formal foundation for our work.

8 http://sbriais.free.fr/tools/abc/

3 CONSTRUCTING ALIGNMENTS

This chapter is based on results published in [498, 500, 495].



BEHAVIOURAL analysis of process models capturing the same business process requires the construction of an alignment between them. We have to identify correspondences between process model elements, in particular between activities of both models. In this chapter, we elaborate on the challenges of constructing an alignment and the techniques to address these challenges. In Section 3.1, we clarify terminology and define basic concepts of an alignment for net systems. Then, Section 3.2 reviews related work. We discuss different types of heterogeneity observed between process models and techniques that aim at avoiding such heterogeneity in the course of model creation. We also include a discussion of work on textual, structural, and behavioural techniques to identify correspondences between process models. This reveals a predominant focus on matching single activities of different models. Complex correspondences

between sets of activities are largely neglected. We address this shortcoming with the ICoP framework in Section 3.3. It proposes a system architecture for the definition of matchers that derive complex correspondences between two process models. The architecture is complemented by a set of basic matching components used to assemble concrete matchers. We evaluate the framework experimentally in Section 3.4. Finally, Section 3.5 concludes this chapter.

3.1 TERMINOLOGY

Alignment of business processes and process models, respectively, has been discussed against diverse backgrounds in the literature. Process models may be aligned, for instance, with business objectives [205, 374, 290, 420, 23], compliance requirements [175, 226, 401, 286, 25, 292, 27], or information on process execution [450, 395, 132]. As a consequence, clarification of the basic concepts of an alignment in our setting is needed. We start by introducing those concepts informally for general process models. Then, we formally define those concepts for net systems.

Basic Concepts

Our interpretation of an alignment is inspired by the alignment of conceptual models as known from the area of data integration and ontology matching [370, 131, 418, 85]. An alignment in this context refers to an association between semantically related entities of data schemas or ontologies. Nevertheless, a common terminology is also missing in this field of research, cf., [333, 224, 153]. For our purpose, we rely on the terminology introduced in [153] and adapt it to the setting of process model alignments.

Figure 12 illustrates the main concepts to discuss alignments of process models. Process models are built of process model elements, such as activities, control flow routing elements, or additional artefacts. We aim at behavioural analysis of alignments, so that we restrict the discussion to alignments that are defined for activities of process models. Figure 12 captures this restricted view by defining a process model as a set of activities. A *correspondence* relates two non-empty sets of activities to each other. Here, all activities of a given set necessarily belong to the same process model. Further, two sets that form a correspondence must relate to distinct process models. For a set of activities in one model, any set of activities of another model that is associated by a correspondence is referred to as a *corresponding* set of activities. We distinguish two types of cor-

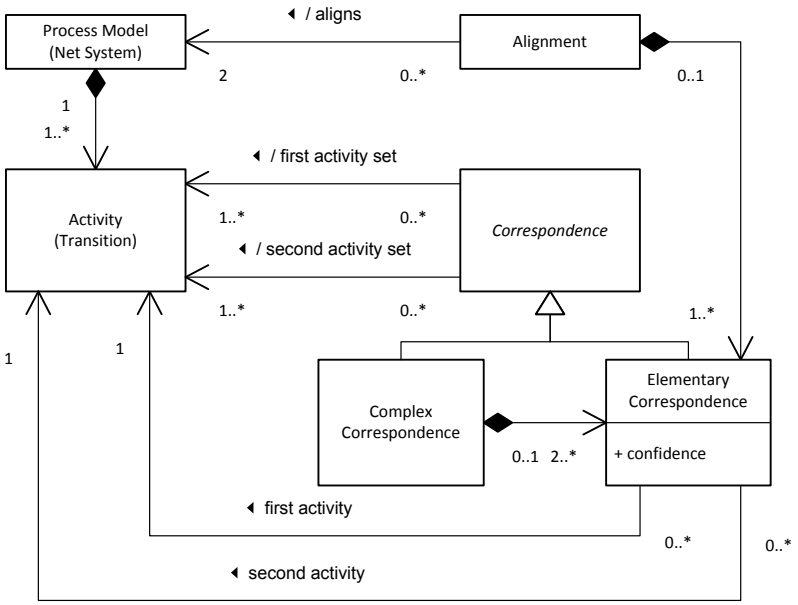


Figure 12: Relations between alignment concepts.

respondences. In the case of *elementary correspondences*, both sets of activities that are associated to each other comprise a single activity. Such a correspondence has a certain *confidence*. In the case of *complex correspondences*, in turn, at least one of the sets comprises two activities. Conceptually, a complex correspondence is formed by multiple elementary correspondences, i. e., the Cartesian product of the associated sets of activities is captured by elementary correspondences. In other words, a certain set of elementary correspondences – the set is maximal with respect to set inclusion – is interpreted as a complex correspondence. Hence, it suffices to base the concept of an alignment on elementary correspondences. An *alignment* is built from a set of elementary correspondences for which the *first activities* all refer to one process model and the *second activities* all refer to another model. Thus, an alignment relates two process models to each other. This relation is induced by the process models from which the activities referenced in the respective correspondences are originating.

Correspondences are a rather generic concept to express a relation between activities. Different semantics may be defined for a correspondence. For instance, a correspondence between two activities may be interpreted as semantic equivalence. In this case, corresponding activities in two process models would refer to exactly the same pieces of work of the original business process. This is rarely the case for correspondences between process models that depict different perspectives on a business process, cf., Section 1.3. A fine-grained analysis of the correspondence semantics is beyond the scope of this work. Hence, we do not

further discuss semantics of correspondences but assume an interpretation driven by the purpose of the alignment.

The procedure of constructing an alignment between process models is called *matching*, tools that support this procedure are called *matchers*. The result of a matching procedure, i. e., a set of elementary correspondences, may also be referred to as a set of *matches*. In the context of our work, we use the term matches solely for relations between characteristics of activities, e. g., there is a match between the labels of two activities. Those matches are at the core of the matching procedure. Once a relation between activities is established, however, we refer to this relation as a correspondence.

Finally, an alignment between data schemas or ontologies is called *mapping* if it is directed [153]. A directed relation is needed to define transformations for the instances of data schemas or ontologies. For instance, a 2:1 complex correspondence between numeric data fields may induce a mapping, such that the sum of two values from one schema corresponds to a single value in the other schema. In contrast, relations on the instance level can be deduced directly from the relations on the model level in our case. A correspondence between two sets of activities of two process models induces a correspondence relation between the instances, i. e., the execution sequences of the process models. That is, the occurrences of the respective activities in an execution sequence of one model correspond to the occurrences of the corresponding activities in an execution sequence of the other model. There is no need to define explicit mappings to transform the instances of process models. Hence, it suffices to focus on undirected alignments to assess behaviour consistency between process models.

Alignments of Net Systems

Having discussed basic concepts of alignments between process models, we formally define those concepts for net systems. We introduced correspondences as a relation between activities of process models. Against the background of modelling business processes with net systems, correspondences between net systems are defined for those elements that represent activities, i. e., transitions. Transitions of net systems may not only represent activities, but may also represent control flow routing elements or may be required for syntactical reasons. However, this is not an issue, as an alignment may be partial in any case.

We capture alignments between net systems by means of a correspondence relation between their transitions and a confidence function. The correspondence relation relates pairs of transitions of two net systems to each other. Based on this relation, the

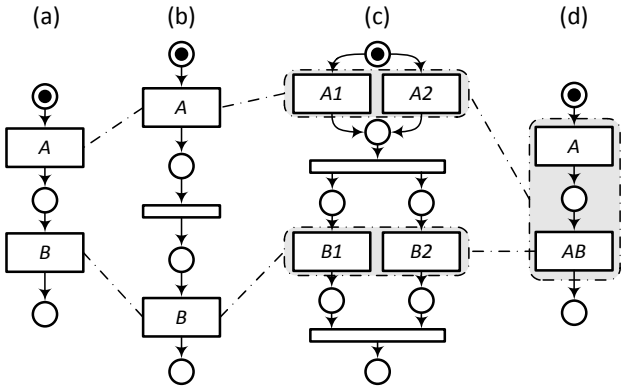


Figure 13: Four aligned net systems.

notion of a correspondence is derived. A correspondence comprises two sets of transitions, for which the Cartesian product of transitions is part of the correspondence relation. The confidence function assigns a confidence value to each entry of the correspondence relation. Further, we say that two correspondences are overlapping, if there is an overlap of the respective sets of transitions in at least one of the net systems.

Definition 3.1.1 (Alignment)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be two net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$.

- A *correspondence relation* $\sim \subseteq T_1 \times T_2$ associates corresponding transitions of both systems to each other.
- Let $T_1' \subseteq T_1$ and $T_2' \subseteq T_2$ be two sets of transitions such that $T_1' \times T_2' \subseteq \sim$. Let T_1' and T_2' be maximal with respect to set inclusion, i. e., $\forall\ t_1 \in (T_1 \setminus T_1')\ [\ (\{t_1\} \times T_2') \not\subseteq \sim\ ]$ and $\forall\ t_2 \in (T_2 \setminus T_2')\ [\ (T_1' \times \{t_2\}) \not\subseteq \sim\ ]$. Then, $c = (T_1', T_2')$ is referred to as a *correspondence* and we also write $T_1' \sim T_2'$.
- A *confidence function* $\zeta : \sim \mapsto [0, 1]$ assigns confidence values between zero and one to single pairs of corresponding transitions.
- A correspondence $c = (T_1', T_2')$ is called *elementary*, iff $|T_1'| = |T_2'| = 1$, and *complex* otherwise.
- Two correspondences $c_1 = (T_1', T_2')$ and $c_2 = (T_1'', T_2'')$ are *overlapping*, iff $T_1' \cap T_1'' \neq \emptyset$ or $T_2' \cap T_2'' \neq \emptyset$.
- A correspondence relation $\sim \subseteq T_1 \times T_2$ is *overlapping*, iff it induces at least two overlapping correspondences. Otherwise, it is *non-overlapping*.
- An *alignment* is a tuple $(\sim, \zeta)$, such that $\sim$ is a correspondence relation and ζ is a confidence function for $\sim$.

We illustrate the introduced concepts with the four net systems depicted in Figure 13. The first alignment between systems (a) and (b) comprises two elementary correspondences between the equally labelled transitions. Between systems (b) and (c), we

observe a complex 1:2 correspondence that associates the transition set $\{A\}$ with the transition set $\{A1, A2\}$. It is represented in the correspondence relation between both systems by two entries, i. e., $A \sim A_1$ and $A \sim A_2$. Between both systems, there is another complex 1:2 correspondence incorporating the sets $\{B\}$ and $\{B1, B2\}$, respectively. The two correspondences are non-overlapping. For systems (c) and (d), there is a complex 2:2 correspondence between the sets $\{A1, A2\}$ and $\{A, AB\}$ and a complex 2:1 correspondence between $\{B1, B2\}$ and $\{AB\}$. In contrast to the aforementioned alignment, the two correspondences between systems (c) and (d) are overlapping. Transition AB in system (d) is part of both correspondences, whereas transition A in system (d) is related only to the set $\{A1, A2\}$ in system (c). Overlapping correspondences may be interpreted as follows. Transitions that are part of the overlap would have to be split up to achieve a clear separation between corresponding sets of transitions. In our example, a *part* of transition AB corresponds to the set $\{B1, B2\}$ in system (c), whereas the remaining *part*, together with transition A, corresponds to the set $\{A1, A2\}$ in system (c).

Using the terminology introduced for relations in Section 2.2, we classify a correspondence relation $\sim$ between two net systems $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$. It is said to be

- *total* from S_1 to S_2, if $\forall\, t_1 \in T_1\, [\, \exists\, t_2 \in T_2\, [\, t_1 \sim t_2\,]]$,
- *functional*, if $\forall\, t_a \in T_1, t_x, t_y \in T_2\, [\, (t_a \sim t_x) \wedge (t_a \sim t_y) \Rightarrow (t_x = t_y)\,]$,
- *injective*, if $\forall\, t_a, t_b \in T_1, t_x \in T_2\, [\, (t_a \sim t_x) \wedge (t_b \sim t_x) \Rightarrow (t_a = t_b)\,]$,
- *bijective*, if it is total in both directions, functional, and injective.

Consider the alignments visualised in Figure 13 and assume that the correspondence relations are defined such that the depicted order of systems is respected. The correspondence relation between systems (a) and (b) is total from (a) to (b), but not vice versa, and functional and injective. For the systems (b) and (c), the correspondence relation is not total in either direction. With (b) as the first system and (c) as the second system of the correspondence relation, it is not functional, but injective. The last alignment is total from (d) to (c), but not vice versa. It is neither functional nor injective.

3.2 MODEL MATCHING

Construction of an alignment between two conceptual models is known as the *matching problem* [153]. This section reviews related work on solving this problem. We start by summarising differ-

ent types of model heterogeneity and strategies to avoid them during the creation of conceptual models, and process models, in particular. Then, we review related work on identifying correspondences between process models. This problem has been extensively addressed in the field of data integration and ontology matching [370, 131, 418, 85], so that we cannot give an exhaustive overview of this work. Instead, we focus on techniques that are relevant for the identification of correspondences between process models. Those are classified according to the aspect of a process model that is considered during matching. We review textual, structural, and behavioural techniques. Behavioural techniques consider the execution semantics of process models. Finally, we turn the focus on complex correspondences.

Model Heterogeneity

Solving the matching problem for a given pair of conceptual models requires coping with different kinds of heterogeneity. The purpose for which a model is created, cf., the drivers of process modelling discussed in Section 1.2, is a common source of model heterogeneity. In addition, heterogeneity may stem from contextual factors, e. g., the background of the human creating a process model [31]. Heterogeneity between conceptual models covers syntactic, terminological, conceptual, or semiotic issues, see [34, 417, 235, 483, 153] for further classifications of model heterogeneity. Semiotic issues refer to different interpretations of models by humans – a problem that cannot be addressed by automated processing. Therefore, we do not further elaborate on this kind of heterogeneity. Regarding the other types of heterogeneity, there is a multitude of approaches that aim at avoiding heterogeneity between process models. In the remainder of this section, we review the most important of these approaches.

Syntactic heterogeneity is observed when conceptual models are defined in different languages. When reviewing process description languages in Section 2.1, we already mentioned several transformations between these languages, e. g., [202, 240, 244, 216, 86, 310, 114, 534, 482]. Most prominently, a potential round-tripping between BPMN collaboration diagrams and BPEL processes has been investigated in several papers [373, 340, 339, 166, 497, 245]. Here, issues that stem from the conceptual differences of a graph-based or a block-based modelling paradigm have been in the centre of interest. The core of this problem, the question if and how an arbitrary process model can be restructured into a well-structured model that is behaviour equivalent, was addressed recently [358, 361]. Despite those differences related to the modelling paradigm, pattern-based evaluations of process description languages received much attention. For in-

stance, most of the languages mentioned in Section 2.1 have been evaluated based on the workflow patterns framework[1] that covers control flow [456], data flow [397], resources [398], and exception handling [399], see [518, 519, 520] for detailed evaluation results. We summarise that syntactic heterogeneity may be bridged by transformations between different languages. However, varying expressiveness and conceptual differences impose serious challenges for these transformations. In the remainder of this work, we neglect this type of heterogeneity and assume all process models to be defined as net systems.

Terminological heterogeneity is observed when different names are used in conceptual models to refer to the same entities of the original, cf., Section 1.1. In process models, terminological heterogeneity may be observed for different kinds of model elements, e. g., activities, roles, and artefacts. Terminological heterogeneity is an issue not only for the creation of alignments between process models. Labelling of process model elements, and especially activities, is known to have a significant influence on the understandability of process models [315, 317]. Hence, it has been advocated to agree on key terms before the creation of process models [391]. Further, there exists a large body of work that aims at labelling support for process models. Conventions such as the verb-object-style [321, 416, 297, 317] and grammatical phrase structures [39] for naming model elements have been suggested. Such techniques require controlled vocabularies of objects and their relations. Object vocabularies may be grounded on ontological concepts, see [188, 162, 442, 169, 490]. The latter, a controlled vocabulary of object relations, has been addressed by semantic classifications of verb phrases [46, 432], categorisations of object relations [268, 319], and the integration of linguistic theories, e. g., speech-acts theory [414], into the process of modelling [76, 8]. Still, all these works require a coordinated creation of process models to obviate terminological heterogeneity. Once process models have been created independent of each other, therefore, this kind of heterogeneity needs to be addressed as part of the matching process.

Conceptual heterogeneity is observed when there are differences in how the domain of interest is modelled. According to [44, 153], there are three reasons for conceptual heterogeneity, differences in coverage, granularity, and perspective.

Differences in coverage are observed when conceptual models describe different parts of the original. In terms of process models, different parts of one or more business processes may be captured. Scoping of process models depends on the scoping of business processes, a question that has been around since the functional breakdown of business functions has been intro-

1 See http://www.workflowpatterns.com for further details.

duced [365] and process orientation emerged as an organisation principle [165, 100, 195, 406]. The question of how to scope process models consistently to avoid differences in model coverage is addressed by top-down modelling methodologies. Such approaches advocate the creation of process models from more abstract descriptions, such as business models and value networks [178, 21, 516, 22, 179, 118], or goal descriptions [229, 242, 261, 182, 115, 23].

Differences in granularity refer to the assumed level of abstraction from the original. Two process models that cover the same part of one or more business processes may comprise a different amount of modelling concepts due to differences in the abstraction level. Many of the techniques discussed to avoid terminological heterogeneity, such as controlled vocabularies and ontologies, along with the aforementioned modelling methodologies also aim at avoiding differences related to the modelling granularity.

Differences in perspective refer to differences that are induced by a certain modelling perspective when creating a conceptual model. Often the modelling perspective is closely attached to the purpose for which a model is created. The often quoted 'Business-IT-Gap' [70, 186, 388] would be an example that causes differences between process models that are conceptual, but cannot be attributed to coverage of the business process or the level of granularity. Such heterogeneity is inherent for models serving different purposes and resolution of the respective differences may negatively affect the adequacy of a process model.

We conclude that differences related to conceptual heterogeneity can be controlled to some extent, but cannot be avoided once process models are created in different contexts and for different purposes. Hence, the construction of an alignment needs to cope with all three kinds of conceptual heterogeneity.

Finally, there have been various attempts to classify differences that relate to conceptual heterogeneity between process models. Differences between views on processes that are captured by object life-cycles are described in [366]. This work also highlights that these differences stem from the independent creation of views for different purposes, as we discussed it for process models in Section 1.2. Differences observed during process evolution have been captured by change patterns [488]. These patterns provide an overview of elementary model differences mostly related to model coverage and granularity. Similar patterns have also been presented in the context of similarity assessment [123] or process enactment [206]. Finally, we presented an overview of difference between process models that capture different modelling perspectives in earlier work [498].

Some of the mentioned heterogeneity issues are addressed by guidelines for the creation of process models [248, 38, 318]. Still, such guidelines either focus on certain aspects only or are defined on a rather abstract level. Hence, they may control some heterogeneity issues, but cannot be assumed to generally avoid heterogeneity between process models.

Textual Matching Techniques

Textual techniques are at the core of most matching approaches. They aim at matching model elements based on their textual description. String-based methods are used to assess the similarity of element labels, which often involves preprocessing of strings. Terms are normalised by considering all characters in lowercase and replacing blanks and diacritics. Then, equality, substring containment, the Hamming distance [196] to count different characters, or the number of equal substrings (n-grams) is used to judge on similarity [153]. Two strings may also be compared using an edit distance. The string edit distance [267] counts the minimal number of atomic character operations (insert, delete, update) needed to transform one string into another. Other measures for string comparison consider the number and proximity of shared characters or the length of common prefixes, such as the Jaro measure [219] or the Jaro-Winkler measure [517]. See [88] for an overview on string distances metrics. These techniques consider strings as a whole. However, strings may be tokenised into a bag of terms [491] to represent them as vectors in a space [402]. Then, each term represents a dimension of the space and the similarity of strings is traced back to the similarity of vectors, e. g., measured by the Cosine similarity [402] or one of the Minkowski distances, see also [153]. The creation of the vector space may reflect different weights for terms. A common measure to assess term weights in a corpus is the TF-IDF scheme [386]. It assigns high weights to terms that occur frequently in one string (TF, term frequency) and rarely in other strings (IDF, inverse document frequency). Still, these techniques are sensitive to linguistic phenomena, such as synonymy and homonymy. Techniques from the field of Natural Language Processing [299] may be applied to overcome this shortcoming. This involves term stemming [364], stop-word elimination [212], or part-of-speech tagging [67]. In addition, external knowledge in the sense of thesauri like WordNet [322] may be leveraged, cf., [173, 346, 72].

Textual techniques are at the core of methods for searching process model collections [136, 473, 127, 530, 129] or service repositories [294, 525, 91, 144, 184], providing modelling support [142, 214, 421, 350], or managing different variants of a

process [330, 123, 271, 494, 258]. Most of these approaches incorporate one of the aforementioned string distances to judge the similarity of element labels. In addition, various approaches leverage synonym relations when comparing element labels on a term basis, e. g., [91, 184, 142, 214, 473, 258, 129]. To this end, virtually all approaches rely on WordNet [322], even though there are differences in the concrete operationalisation. For instance, synonym dependencies may be weighted relative to the overall number of senses of the respective terms [142]. Part-of-speech tagging has been applied to refactor labels of process model elements [266] or for aggregating element labels [422]. The latter work uses the MIT Process Handbook [297], which can be seen as an ontology for business processes. The MIT Process Handbook spans several business domains, e. g., sales or production. Although these techniques do not directly contribute to an identification of correspondences, they allow for a normalisation of labels before applying further matching techniques. Structural characteristics of ontological annotations of process models have also been exploited to identify correspondences [142, 68]. Further, ontologies have been applied to identify correspondences between service descriptions, mainly by relating input and output specifications to each other [344, 47]. Matching based on vector spaces has been used for process models and service descriptions in [294, 285, 214]. To this end, terms of the textual description of process model elements [214] or explicit feature annotations [285], e. g., roles and IT systems, are interpreted as dimensions of the vector space. Then, standard means for judging the similarity of vectors are used to conclude on activity correspondences.

Structural Matching Techniques

Structural matching techniques exploit the graph structure of process models. In particular, techniques that built upon the notions of a maximum common sub-graph isomorphism and the graph edit distance have been used, see [74] for an overview of graph matching techniques. The maximum common sub-graph (MCS) problem refers to the identification of a maximal sub-graph of two graphs. The graph edit distance (GED) defines the minimal number of atomic graph operations (substitute node, insert/delete node, (un)grouping nodes, substitute edge, insert/delete edge) needed to transform one graph into another [75]. Both, the MCS and the GED problem, are closely related for a certain class of cost functions for the edit operations [73]. Unfortunately, both problems are NP-hard [168]. This leads to the application of search algorithms, such as the A^* search [201], or heuristics.

Based on the graph edit distance, a similarity score may be computed for a pair of aligned process models [128, 127]. Here, only model elements that are part of correspondences are considered to be substituted, while the quality of substitution is based on the similarity of the elements. The quality of the alignment is then measured relative to a hypothetical ideal alignment. The latter relates each element and each flow in one process model to an element or a flow in the other process model with an optimal substitution quality. Due to the computational complexity of the GED problem, different heuristics have been developed to identify the optimal alignment between two process models [91, 128]. Matching approaches based on the graph edit distance may be extended by considering characteristics of process models, such as different element types [325]. Distinguishing different element types in process models, gives rise to further matching techniques. A process model may be reduced by reduction rules that consider different element types to assess equivalence or subsumption with another model [285]. Other approaches customise the graph edit distance by considering high-level and hierarchical change operations that go beyond simple insertion, substitution, and deletion [271, 256]. To this end, relations between activities are exploited [271] or the process model is structurally decomposed into fragments [256].

Besides graph matching, there are several structural strategies to influence the similarity of process model elements to detect correspondences between them. Elements may be classified according to structural patterns depending on the cardinality constraints of their incoming and outgoing flows [530]. These features influence the similarity when comparing two model elements. Contextual similarity as proposed in [473, 129] exploits the share of directly preceding or succeeding elements that show a high similarity. Following this idea, similarity flooding introduced for matching data schemas [311] has been applied to match process models [294]. It first judges the similarity of model elements with a textual measure applied to the element labels. Then, the similarity of elements is iteratively updated according to the similarity of adjacent elements until a fixpoint is reached [294].

Behavioural Matching Techniques

Behavioural matching techniques leverage the behaviour of process models in terms of their execution semantics. Here, the type of assumed behavioural semantics has to be considered, may it be trace-based or using a labelled transition system, see Section 2.2. Moreover, the chosen process description language affects behavioural matching techniques. A close relation between

syntax and execution semantics allows us to draw conclusions on execution semantics by looking at the model structure only. Most business process description languages, however, are not grounded on a small set of semantically orthogonal concepts, cf., the Orthogonality Principle of model formalisation [437]. Several techniques aim at addressing this issue by transforming process models in a normal form before applying structural means to assess similarity. An example would be the splitting of activities that represent synchronous service calls into a sending activity and a receiving activity [91].

Under the assumption of trace semantics, similarity measures may be defined directly on the language of a process model, i. e., the set of all traces. The size of the intersection of the languages of process models relative to the overall number of traces would be a straight-forward example for such a measure [136]. Following on the idea of the string edit distance, edit distances have also been defined for the behaviour of a process model in terms of its language, an automaton encoding the language, or an n-gram representation of the language [525]. A fine-granular variant of the latter measure considers not only all n-grams needed to define the language, but also their cardinalities to control the impact of control flow loops on the measure [523]. Experimental results obtained with these measures in the context of service retrieval can be found in [522, 524, 523]. Once the behaviour of a process model is captured by a transition system, the degree to which two systems simulate each other was proposed as a similarity measure [427, 330]. Given a similarity measure for state transitions, similarity of states is evaluated iteratively by considering the similarities of neighbouring state transitions and states. This approach terminates after a fixpoint or an iteration boundary is reached.

Behavioural abstractions that capture only dedicated behavioural aspects may also be leveraged for the identification of correspondences between process models. Behavioural relations that are defined over elementary activities, e. g., order and exclusiveness, provide a means to enrich structural matching with behavioural information [144]. Those behavioural details may be closely related to the model structure, as shown for BPEL processes in [144]. Other works uses causal footprints as an approximation of the process model behaviour to assess similarity [473]. Causal footprints capture causal dependencies for activities by defining, for each activity, a set of causal predecessors and a set of causal successors.

On the Identification of Complex Correspondences

Once we reviewed basic matching techniques, the identification of complex correspondences deserves further discussion. Diverging modelling purposes as discussed in Section 1.2 are likely to cause conceptual heterogeneity. Once different levels of abstraction are assumed, an alignment comprises complex correspondences. The importance of complex correspondences is further witnessed by the aforementioned classifications of differences between process models [366, 488, 123, 206, 498]. All of these classifications contain patterns that capture refined activities, activity fragmentation, or activity aggregation. In Section 3.1, we discussed that, conceptually, complex correspondences are built from elementary correspondences. Nevertheless, complex correspondences are typically identified directly. An identification of all overlapping elementary correspondences that form a complex correspondence is hardly feasible.

The identification of complex correspondences between process models has been largely neglected in the literature. Virtually no work uses the introduced matching techniques to find correspondences beyond elementary correspondences in a non-overlapping alignment. Complex correspondence have only be addressed in a narrow language-specific setting. Splitting up BPEL activities that represent synchronous service calls into two activities, see [91], is an example for such a heuristic. To the best of our knowledge, there has not been any attempt to address the issue of complex correspondences between process models in general way.

Unfortunately, there has also been a predominant focus on elementary 1:1 correspondences in the schema and ontology matching community, such that *'1:n and n:m mappings [..] are currently hardly treated at all'* [370]. One of the rare exceptions is the iMAP system [120]. iMap proposes to explore the space of potential mapping expressions between arbitrary groups of elements using different search heuristics. Once potential element mappings are identified, the similarity of target entities is analysed to select the final mapping. iMAP searchers exploit the value distribution of instance data and also take domain knowledge, such as domain constraints, into account. Other work on the identification of complex correspondences between data schemas relies on the discovery of characteristics for instance data and the application of domain ontologies that describe expected data values [529]. Further, the application of correlation mining techniques was proposed in the DCM framework [203]. This framework mines web query interfaces to identify grouping attributes, i. e., attributes that tend to be co-occurring in web interfaces. This knowledge is exploited to mine negative correla-

tions between groups of attributes. Those hint at potential complex correspondences. There are further schema matching approaches that retrieve complex correspondences by just applying a static similarity threshold for the selection of correspondences, e. g., the Cupid matcher [293]. Given a similarity matrix for all model elements, various combinations for a single element may show similarity values above the threshold, such that complex correspondences are derived. However, such an approach does not hint at strategies that are used to identify complex correspondences, as it assumes this knowledge to be already encoded in the similarity matrix.

We summarize that the few existing approaches for identifying complex correspondences between data schemas rely on instance data and external knowledge. As the former is not always available for process models, these techniques cannot be transferred to the setting of process models directly.

3.3 THE ICOP FRAMEWORK

The review of existing matching techniques reveals that there is virtually no work available on the identification of complex correspondences between process models. Further, even in the field of schema and ontology matching there has been a predominant focus on elementary correspondences, despite a few notable exceptions. We argued that complex correspondences, however, are essential for aligning process models for consistency analysis. To address this issue, this section introduces the ICoP (Identification of Complex Correspondences between Process Models) framework. It defines a system architecture for the definition of matchers that derive complex correspondences between two process models. Besides the architecture, it also introduces a set of basic matching components used to assemble concrete matchers. These components focus on the identification of complex 1:n correspondences that are non-overlapping, as those correspondences can be expected to occur frequently between process models. Still, we also discuss how the components may be adapted to consider complex n:m correspondences.

The ICoP framework is independent of any notion of a process model or process description language. All discussed matching components work purely on the textual and structural level. Although we use net systems for illustration purposes, we do not stick to the net system terminology but use the generic terms process model and activity. In the remainder of this section, we first present the overall architecture of the ICoP framework. Then, we elaborate on four different types of components in detail and present exemplary realisations.

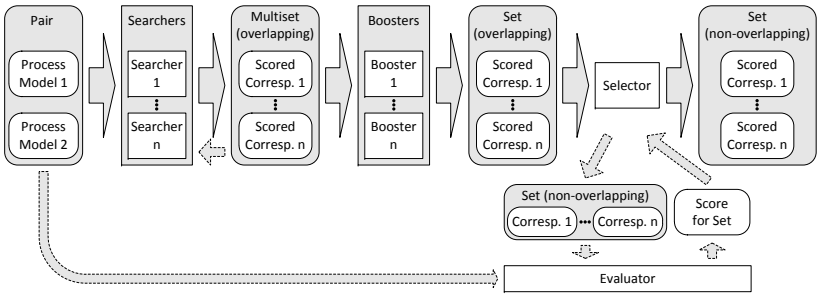


Figure 14: The architecture of the ICoP Framework.

Architecture

The overall architecture of the ICoP framework is motivated by the observation that the number of complex correspondences between two process models is potentially large. It is not feasible to explore all possible correspondences exhaustively. Instead, the ICoP framework proposes a multi-step approach, which is illustrated in Figure 14. Given two process models, *searchers* extract potential correspondences based on different similarity measures and heuristics for the selection of activities. The result of the search stage is a multiset of correspondences – multiple searchers may identify the same potential correspondences. Each correspondence is assigned a *confidence score* that reflects the quality of the relation. It results from the scoring function implemented by the searcher to select potential correspondences. A searcher may use the knowledge about potential correspondences that have been identified by other searchers already.

After completion of the search stage, the scored potential correspondences are conveyed to *boosters*. These components *boost* the correspondences that are returned by the searchers using heuristics to aggregate or remove correspondences, or adapt the score of a correspondence. As part of the boosting stage the *multiset* of potential correspondences is transformed into a *set* of potential correspondences by aggregating those that have been identified by multiple searchers.

Then, a *selector* builds up the actual alignment from the set of potential correspondences. It selects the best correspondences from the set of potential correspondences. The selector components presented later ensure that the constraint of non-overlapping correspondences is satisfied. The selection of the best correspondences is guided by two kinds of scores. On the one hand, the individual scores of the potential correspondences are exploited. On the other hand, an *evaluator* is utilised, which assigns a single *alignment score* to an alignment. An evaluator may use knowledge about the original process models to compute this score. The selection is an iterative process. In each itera-

tion, the selector selects a set of correspondences, the evaluator computes the score for this set, upon which the selector either decides to modify the set of correspondences and continue the selection, or to complete the selection. Once the selection procedure completes, the selector produces the final alignment between activities of the process models.

The presented architecture is inspired by the structure of the iMAP system [120], which we discussed in the previous section. It introduces the idea of searching the space of potential complex correspondences. Besides the differences in the implementation of the searcher components – those of the iMAP rely on instance data – there is another conceptual difference of the ICoP architecture and the iMAP system. iMAP searchers derive textual, numeric, or structural mapping expressions instead of pure element correspondences. We elaborated in Section 3.1 on why the detection of mapping expressions for transforming instances is of minor importance in the context of process models.

Searchers

Searchers identify potential correspondences between two process models along with a score, which represents the quality of the correspondence. We denote a correspondence between two activity sets A_1 and A_2 that holds with confidence c by a tuple (A_1, A_2, c). This is a short-hand notation for the notions introduced in Section 3.1. If the respective entries of the correspondence relation $(a_1, a_2) \in (A_1 \times A_2)$ have different confidence values assigned, we assume the score c to represent the arithmetic mean of these confidence values.

As discussed in the previous section, identification of a correspondence may be based on various aspects, including the labels or descriptions of activities and structural or behavioural relations between those elements. We introduce four searchers that have been implemented in the ICoP framework. Two of them focus on elementary correspondences, whereas the other two search for complex 1:n correspondences. The latter may be adapted to search for n:m correspondences at the expense of increased computational complexity.

Similar Label Searcher. The purpose of this searcher is the identification of 1:1 correspondences based on a high syntactic similarity of activity labels. It computes the Cartesian product of activities of two process models and selects all pairs for which the string edit similarity of their labels is above a threshold. For two strings s_1 and s_2 the string edit similarity is defined as $sim(s_1, s_2) = 1 - (ed(s_1, s_2) / \max(|s_1|, |s_2|))$ with $ed(s_1, s_2)$ as the string edit distance [267]. String edit similarities can be seen as rather strict criteria when applied to compare activity labels,

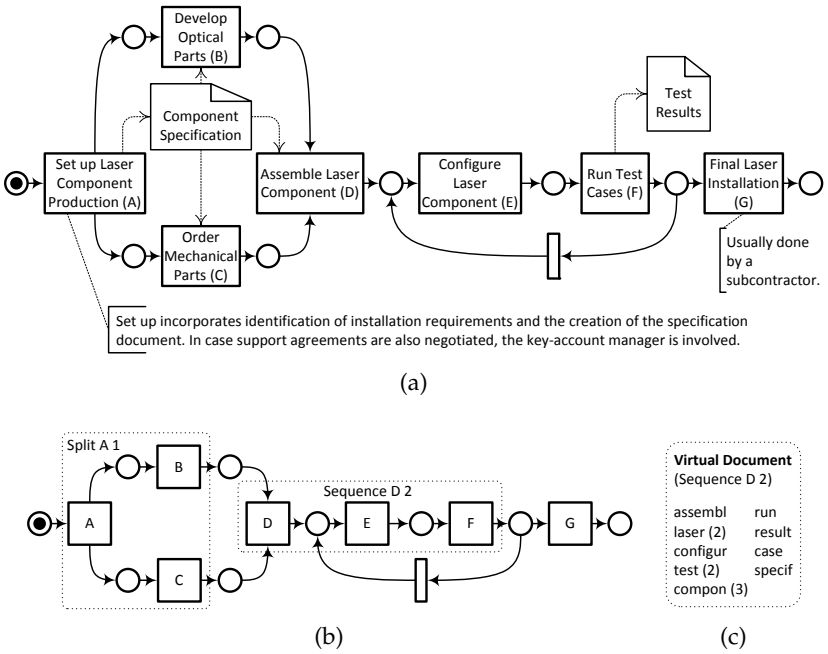


Figure 15: (a) Process model with annotations, data objects and data associations; (b) the control flow structure of (a) along with two groups as considered by the Distance Doc Searcher, (c) virtual document for the group 'Sequence D 2'.

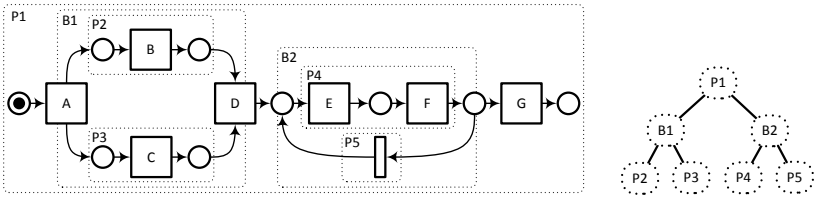
not only words of such labels. They neglect different orders of words and linguistic phenomena such as synonymy. Hence, the potential matches are identified with high confidence, such that the initial score for these matches is set to one by the searcher. The run time complexity of this searcher depends solely on the number of activities of the respective process models.

Distance Doc Searcher. This searcher follows a two step approach to identify potential 1:n correspondences. First, activities of both process models are grouped heuristically. Second, the similarity between such a group of activities in one model and all single activities in the other model is assessed. Although this approach yields potential 1:n correspondences, groups of activities of both models may also be exploited. Still, this would increase the run time complexity of the searcher due to the implied combinatorial problem. We illustrate the two steps using the process model depicted in Figure 15a. Assuming that it is more likely that activities that are closer to each other should be in the same group, we use the graph distance to group activities. The graph distance between two activities is the number of edges on the shortest path from one activity to the other [122]. Given a base activity and a distance, we look for four types of groups:

- Sequences, which are determined by a base activity and the activities on a directed path of the given length (distance) from the base activity.
- Splits, which are determined by a base activity and the activities that can be reached from the base activity and that are within the given distance. The base activity can or cannot be considered in such groups, depending on whether the routing is modelled explicitly by a control flow node.
- Joins, which are determined by a base activity and the activities from which the base activity can be reached and that are within the given distance. The base activity can or cannot be considered in such groups.
- Others, which are the groups that consist of all activities that are within the given distance of a base activity (not considering the direction of edges). In contrast to sequences, these activities are not necessarily on a path.

For the example model depicted in Figure 15a, Figure 15b shows the control flow structure of the model along with two example groups. The group 'Split A 1' is built by taking activity A as base activity and by exploring all activities that may be reached with a graph distance of one, i. e., activities B and C. Here, we compute the graph distance solely based on activities. Therefore, only flows between transitions and places are counted when computing the graph distance between transitions of the net system. The example group 'Split A 1' illustrates the case, in which the base activity is part of the group. Besides, the searcher would also consider the group consisting of solely activities B and C to account for the fact that the control flow routing is not modelled with a separate node. The second example group is of type sequence, takes activity D as base activity, and considers all activities that are within a graph distance of two. The Distance Doc Searcher identifies all groups of activities in a process model by taking each of the activities as a basis and creating each possible type of group for each possible graph distance value. A maximum distance value is set as a parameter.

Once groups of activities have been identified, their similarity needs to be assessed. Besides labelled activities, a process model may comprise additional information on, for instance, processed data objects and textual annotations, as illustrated in Figure 15a. To take such information into account, we use *virtual documents* to score the similarity of activities or groups thereof. Virtual documents have been introduced for information retrieval [487], and later been applied to identify alignments of ontologies [369]. A virtual document of an ontology node consists of the words from all textual information that is related to that node. Given two virtual documents, their similarity is computed based on their distance in a vector space as discussed in the previous section. This



(a) (b)

Figure 16: (a) Control flow structure of the model in Figure 15a along with RPST fragments, (b) the corresponding RPST.

idea has already been transferred to annotated process models to answer user queries against a model repository [214]. In our setting, a virtual document for an activity consists of the terms that are derived from the activity label and, if this information is available, the labels of the roles that are authorized to perform the activity, the assigned input and output data objects, and a textual description of the activity. These terms are preprocessed with the techniques described earlier, such as filtering of stopwords [212] and term stemming [364]. The preprocessed terms represent the dimensions of the vector space. Further, we rely on the TF-IDF [386] scheme for weighting terms. For a group of activities, the virtual document is derived by joining the documents of the respective nodes. Creating virtual documents from process models can be seen as the reverse operation of generating a textual representation of a conceptual model [16, 194], even though virtual documents are unstructured. Figure 15c illustrates the terms of the virtual document for the group 'Sequence D 2' along with their numbers of occurrences. These terms originate from activity labels and the labels of two data objects. All terms have been preprocessed. The run time complexity of this searcher is influenced by one parameter besides size and structure of the models – the maximal graph distance value for grouping activities.

Fragment Doc Searcher. This searcher resembles the distance doc searcher, except for the strategy for grouping activities. We leverage a structural decomposition technique, introduced as the Refined Process Structure Tree (RPST) [481, 359]. Later, we formally define the RPST for net systems when computing behavioural relations from a process model. At this point, however, we stick to an informal description of the characteristics of this decomposition technique. The RPST parses a process model into a hierarchy of fragments with a single entry node and a single exit node. Figure 16a depicts these fragments for the process model given in Figure 15a. The fragments are defined in such a way that they do not overlap. Consequently, they form a tree-structure, the RPST, which is illustrated in Figure 16b. We leverage the hierarchy of fragments to select groups of activities that

will be considered by the searcher. Starting with the leaf fragments of the RPST, the tree is traversed upwards up to a height that is given as a parameter. For the example in Figure 16b and a height threshold of two, the searcher considers the fragments B1, B2, P2, P3, P4 and P5. For all traversed fragments, the searcher creates two virtual documents. One document comprises all activities contained in the fragment. The second document contains all these activities except for those that are boundary nodes of the fragment. Again, implicit or explicit modelling of control flow routing nodes is the motivation for considering both cases. The similarity of the virtual documents is assessed using the vector space approach as introduced for the Distance Doc Searcher. Therefore, this searcher also considers primarily 1:n correspondences. Although it is possible to compare all groups of activities in both models, the combinatorial problem would impact the run time complexity. Apart from that, the run time complexity of this searcher mainly depends, besides size and structure of the models, on the height up to which the RPST is traversed upwards for identifying groups of activities.

Wrapup Searcher. This searcher resembles the aforementioned Similar Label Searcher. It also aims at deriving potential 1:1 correspondences by analysing the string edit similarity for the labels of activity pairs. In contrast to the Similar Label Searcher, this searcher traverses only activities that are not part of potential correspondences retrieved by other searchers. The Wrapup Searcher is run after all other searchers. Also, the threshold for the similarity of two activity labels is set to a lower value than for the Similar Label Searcher.

Boosters

After potential 1:1 and 1:n correspondences have been identified, the multiset of scored correspondences is propagated to a set of boosters. We implemented the following four boosters as part of the ICoP framework.

Cardinality Booster. This booster reduces the *multiset* of potential correspondences to a *set* by aggregating the confidence scores for potential correspondences that associate the same (sets of) activities to each other. Two correspondences (A_1, A_2, c_1) and (A_3, A_4, c_2) with $A_1 = A_3$ and $A_2 = A_4$ are replaced by a correspondences (A_1, A_2, c_a). We define the aggregated confidence score as $c_a := c_1 + (1 - c_1) \cdot c_2$. The first score is increased by the second score relative to its current value. This operation is symmetric and can iteratively be applied if more than two scores need to be aggregated.

Subsumption Booster. The idea behind this booster is that a 1:n correspondence (A_1, A_2, c_1) may subsume another correspond-

ences (A_3, A_4, c_2), such that $A_3 \subset A_1$ or $A_4 \subset A_2$. Similarity scoring based on the cosine angle in vector spaces tends to favour documents consisting of a small number of terms [222]. Large documents yield a large dimensionality of the representing vector, so that the scalar product between such a vector and any other vector tends to be small. As a consequence, the subsumed correspondences will have higher initial confidence scores on average. To countervail this effect, we boost a 1:n correspondence, if it subsumes other correspondences. If the correspondence (A_1, A_2, c_1) subsumes the correspondence (A_3, A_4, c_2), its confidence score is increased relative to the current value, such that $c_1 := c_1 + w_s \cdot (1 - c_1) \cdot c_2$ with $0 \leqslant w_s \leqslant 1$ as a weighting factor.

Tree Depth Ratio Booster. In contrast to the aforementioned boosters, this booster considers solely a single correspondence. It boosts the confidence score of a correspondence, if the activities show a certain structural property that is evaluated based on the RPST. Given a correspondence (A_1, A_2, c_1), we determine the Lowest Common Ancestors (LCAs) of A_1 and A_2 in the RPST of the respective process model, denoted by $lca(A_1)$ and $lca(A_2)$. Let $maxDepth_1$ and $maxDepth_2$ be the maximal depths of a fragment in the RPSTs of the two process models. Then, we determine two ratios by relating the depth of the LCA to the maximal depth of the tree, $r_1 = lca(A_1)/maxDepth_1$ and $r_2 = lca(A_2)/maxDepth_2$. The confidence score of a correspondence is boosted, if the average of the two ratios is above a threshold. This indicates that both LCAs are relatively low in the tree, which is interpreted as a hint for a good quality of the correspondence. The underlying assumption is that activities that form a correspondence are likely to be structurally close. If the threshold is reached, the confidence score of the correspondence is increased according to the average of the two ratios, relative to the current score value, i. e., $c_1 := c_1 + (1 - c_1) \cdot w_r \cdot 0.5 \cdot (r_1 + r_2)$ with $0 \leqslant w_r \leqslant 1$ as a weighting factor.

Distance Ratio Booster. This booster also considers single correspondences. Here, the structural property that is evaluated relates to the graph distance. For each of the sets of activities A_1 and A_2 of a correspondence (A_1, A_2, c_1), we determine the maximal distance between two activities of this set. For each activity, we compute the distances from and to all other activities, and select the minimal distance. Then, the maximal value of all these minimal distances is chosen, denoted by $maxDist(A_1)$ and $maxDist(A_2)$, respectively. These values provide a measure for the spread of the activities of the correspondence. Let $maxDist_1$ and $maxDist_2$ be the maximal distances that can be observed between any two activities that are connected by a path in the two process models. Those values provide a measure

for the spread of all activities in the process model. Then, we define two ratios $r_1 = 1 - (maxDist(A_1)/maxDist_1)$ and $r_2 = 1 - (maxDist(A_2)/maxDist_2)$. If the average of both ratios is above a threshold, the confidence score of the correspondence is increased accordingly, i. e., $c_1 := c_1 + (1 - c_1) \cdot w_d \cdot 0.5 \cdot (r_1 + r_2)$. Again, $0 \leqslant w_d \leqslant 1$ is a weighting factor.

Selectors

Once the confidence values of correspondences have been adapted by boosters, a selector extracts an alignment from the set of scored potential correspondences. As mentioned before, we require an alignment to satisfy the constraint of non-overlapping correspondences in the ICoP framework. However, this implies that the selection process is a computationally hard problem, which is independent of the different notions of quality for an alignment employed by the selectors. Due to overlapping potential correspondences, finding an alignment with maximal quality is an optimisation problem that may be addressed by search algorithms, such as the A*-algorithm [108]. Those algorithms guarantee to find the optimal solution. However, experiments on matching process models in a similar context revealed that results obtained by a greedy search strategy are close to those obtained with an exhaustive search [127]. Therefore, our selectors follow a greedy or 1-look-ahead strategy. The ICoP framework consists of the following selectors.

Correspondence Similarity Selector. This selector selects an alignment comprising non-overlapping correspondences solely based on their confidence scores. The correspondence with the highest confidence is selected – if there are multiple correspondences with an equal score, one is randomly chosen – and all correspondences that are overlapping with the selected one are removed and not further considered. The alignment is constructed iteratively until the highest confidence for a potential correspondence is below a given threshold. In addition to this greedy strategy, we also implemented a 1-look-ahead strategy. It optimises the score for the succeeding iteration in case there are multiple correspondences with equal confidence scores.

Alignment Similarity Selector. This selector neglects the scores assigned to potential correspondences and relies solely on the score for a (partial) alignment as provided by an evaluator. Correspondences are iteratively selected. In each step, we select the correspondence leading to the maximal score for the alignment. If multiple correspondences meet this requirement, one is chosen randomly. The procedure terminates once the alignment score cannot be increased further. Again, we implemented this greedy strategy and a 1-look-ahead variant. The latter selects the

correspondence that leads to the maximal alignment score in the succeeding iteration.

Combined Selector. This selector uses both, confidence scores of correspondences and alignment scores as provided by an evaluator. In a first step, the highest confidence score among the correspondences is determined. All potential correspondences that have been assigned this value are then selected for the alignment. In case this is not possible due to overlapping correspondences, the selection is conducted randomly. In a second step, the alignment is iteratively extended with the correspondence that maximises a combined score. This score is a weighted average of the confidence of the correspondence and the alignment score as computed by the evaluator. The procedure terminates if the combined score cannot be increased any further. Again, the second step of the selection process has been implemented as a greedy and as a 1-look-ahead strategy.

Evaluators

A selector can use an evaluator to score alignments. Given a (partial) alignment, the evaluators return a single score for the quality of the alignment. Different notions of quality can be considered. Within the ICoP framework, we implemented the following two evaluators.

Graph Edit Distance Evaluator. This evaluator scores a given alignment based on the graph edit distance [75, 74] of the two original process models that is induced by the correspondences. As discussed in Section 3.2, the graph edit distance can be leveraged to define a similarity score [128, 127]. The Graph Edit Distance Evaluator computes this similarity score and returns it as a measure of the quality of the given alignment.

Path Relation Evaluator. This evaluator scores a given alignment based on whether the path relations are preserved for the activities of a pair of correspondences of the alignment. Let $C_1 = (A_1, A_2, c_1)$ and $C_2 = (A_3, A_4, c_2)$ be correspondences. Then, we derive the number of preserved paths $pre(C_1, C_2) = |\{(a_1, a_2, a_3, a_4) \in (A_1 \times A_2 \times A_3 \times A_4) \mid p(a_1, a_3) \Leftrightarrow p(a_2, a_4)\}|$, with $p(x, y)$ being a predicate that denotes the existence of a path from activity x to activity y. The score for the pair of correspondences C_1 and C_2 is defined as $s(C_1, C_2) = pre(C_1, C_2) \,/\, |A_1 \times A_2 \times A_3 \times A_4|$. The score for the alignment is computed by iterating over the Cartesian product of correspondences and computing the average of their scores.

3.4 EXPERIMENTAL EVALUATION

The previous section introduced the ICoP framework to identify complex correspondences between process models. The architecture of the ICoP framework is a generic schema for the definition of matching components that may be reused in multiple matchers. In this section, we report on an experimental evaluation of our framework. In this experiment, we aimed at constructing alignments between process models that (1) capture similar processes in different organisations and (2) represent reference processes and their implementations. These use cases differ from the one primarily investigated in this work – a common process is captured for different purposes. Still, based on the experiment, we are able to draw conclusions on the overall performance of the ICoP framework.

Using the components introduced in the previous section, we assembled matchers and compared the correspondences identified by them with the correspondences that process analysts found in a collection of 20 pairs of process models. Three pairs have been taken from a merger in a large electronics manufacturing company. Each of these pairs represents two processes that have to be merged. 17 pairs have been taken from Dutch municipalities. Each of these pairs represents a standard process[2] and an implementation of this standard process by a municipality. All process models have been available as net systems. Each system from the collection has, on average, 31.1 nodes, with a minimum of 9 nodes and a maximum of 81 nodes for a single model. The average number of arcs pointing into or out of a single node is 1.2 and the average number of words in the label of a single node is 2.8. For the 20 process model pairs, process analysts determined a total of 520 elementary correspondences. Of these 520 elementary correspondences, 221 were part of a complex correspondence. However, the distribution of these complex correspondences in our model collection shows a high variation. For instance, for three out of 20 model pairs – the model pairs from the merger – more than 90% of the elementary correspondences are part of complex correspondences. In turn, six out of 20 model pairs show only elementary correspondences.

We evaluated the performance of the matchers in terms of precision and recall [29]. Precision is the fraction of found elementary correspondences that that is correct, i. e., these correspondences have also been found by process analysts. The recall is the fraction of correct elementary correspondences that is found. The F-Score combines precision and recall in one value. We also computed the Overall score, an alternative metric

2 Documentair structuurplan: http://www.model-dsp.nl/

that has specifically been developed for measuring the quality of schema matches [130]. All metrics are defined for elementary correspondences, so that complex correspondences are implicitly covered. Still, we also distinguished the recall for elementary correspondences that are part of any complex correspondences and those that are not. This allows for conclusion on the ability of the matchers to detect complex correspondences. All metrics are based on the following sets.

$\mathcal{C}_h$	Elementary correspondences identified by process analysts.
$\mathcal{CC}_h \subseteq \mathcal{C}_h$	Elementary correspondences that are part of a complex correspondence.
$\mathcal{CE}_h := \mathcal{C}_h \setminus \mathcal{CC}_h$	Elementary correspondences that are not part of a complex correspondence.
$\mathcal{C}_m$, $\mathcal{CC}_m$, $\mathcal{CE}_m$	Analogously defined sets of correspondences that are identified by matcher m.

Based on these sets, the metrics for the evaluation of matchers are defined as follows.

$$\begin{aligned}
\text{precision} &:= |\mathcal{C}_m \cap \mathcal{C}_h|/|\mathcal{C}_m| \\
\text{recall} &:= |\mathcal{C}_m \cap \mathcal{C}_h|/|\mathcal{C}_h| \\
\text{recall-elementary} &:= |\mathcal{CE}_m \cap \mathcal{CE}_h|/|\mathcal{CE}_h| \\
\text{recall-complex} &:= |\mathcal{CC}_m \cap \mathcal{CC}_h|/|\mathcal{CC}_h| \\
\text{F-score} &:= 2 \cdot (\text{precision} \cdot \text{recall})/(\text{precision} + \text{recall}) \\
\text{Overall} &:= \text{recall} \cdot (2 - 1/\text{precision})
\end{aligned}$$

For our evaluation, we created five matchers within the ICoP framework.

Baseline Matcher. This matcher realises the greedy graph matcher presented in [128] and consists of a Wrapup Searcher, a Graph Edit Distance Evaluator, and a Mapping Similarity Selector. This matcher identifies solely elementary correspondences and does not consider potential complex correspondences. Still, it finds these correspondences with high precision and recall. Therefore, we use it as a baseline benchmark for our framework, which focusses on improving results with respect to complex correspondences.

Matcher A. This matcher comprises all presented searchers and a Look Ahead Match Similarity Selector. It demonstrates the pure performance of our searchers.

Matcher B. This matcher extends Matcher A by incorporating all boosters introduced as part of the ICoP framework. This matcher illustrates the impact of boosters on the matching process.

Matcher C. This matcher consists of all searchers, but in contrast to Matcher A, it takes the evaluation of (partial) alignments

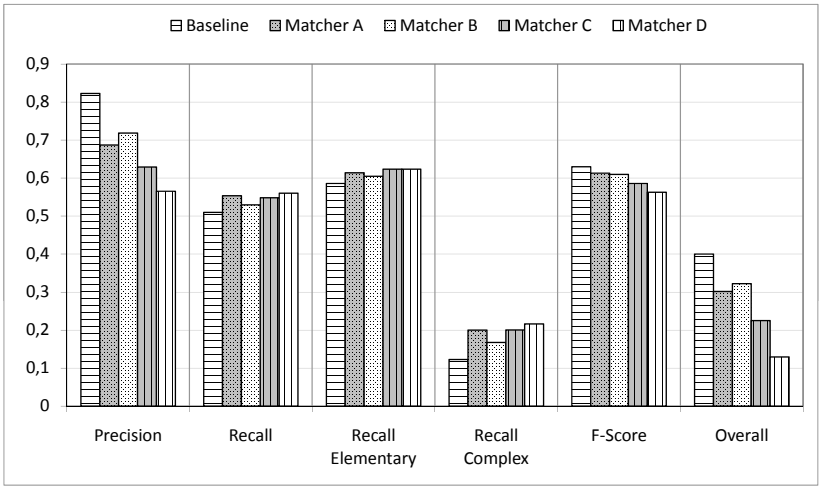


Figure 17: Metrics derived for the ICoP matchers.

into account when selecting an alignment. It relies on the Path Relation Evaluator and a Look Ahead Combined Selector.

Matcher D. This matcher consists of all searchers and evaluates partial alignments, but uses another evaluator than Matcher C, i. e., a Graph Edit Distance Evaluator.

Matchers A and B, as well as C and D, are similar. The first pair of matchers relies on the confidence scores assigned to potential correspondences to build up the alignment. The second pair of matchers uses a combined approach that also utilises the scores derived by evaluator components.

Figure 17 depicts the results of applying these five matchers to all 20 model pairs. It shows the precision, recall, F-Score, and Overall for one specific configuration of each matcher that maximises the F-score for the whole model collection. It also highlights the recall for elementary and complex correspondences separately. Although the Baseline Matcher is not meant to detect complex correspondences, it returns some. Several of the identified elementary correspondences are actually part of a complex correspondence.

The results show that the architecture works. It has an adaptable modular setup and we have been able to reproduce the results obtained by the more rigid matcher presented in [128]. The results also show that the architecture can be used to produce matchers that detect complex correspondences and that their recall is better than that of our baseline matcher. Unfortunately, the complex correspondences improved recall at the expense of precision, leading to an F-Score that is slightly lower than that of the baseline matcher. The comparison of the results for Matcher A and Matcher B reveals that the application of boosters increased the precision, at the cost of decreased re-

call. Similarly, the Path Relation Evaluator in Matcher C led to a better precision than the Graph Edit Distance Evaluator in Matcher D, which, again, is traded for recall to a certain extent.

We observed a large difference between the results with respect to the origin of the process models. As indicated, three model pairs stem from a comparison in a merger, while 17 originate from a comparison of standard processes to their implementations. The results for the merger model pairs were much worse – the F-Score has been around 0.3 – than the results for the standard process comparison pairs. Qualitative analysis of the results revealed that the modest performance of the matchers for the merger scenario was caused by heterogeneous activity labels. We conclude that, in order for the current matchers to work, there must be some level of similarity between activity labels of corresponding activities.

The models used in the experiment are different from those that we expect in the context of behaviour consistency analysis. In particular in the merger scenario, the models for which we construct an alignment capture *different* business processes, i. e., different originals have been abstracted by the models. Hence, a rather high terminological heterogeneity is no surprise for these models. They are grounded on completely different vocabularies of concepts as established in the organisations that should be merged. This raises the question to which extent terminological heterogeneity can be expected to hold for process models that capture a common process for different purposes. The focus on a common process within one organisation suggests that the terminological heterogeneity between such models can be expected to be less strong than observed for the merger scenario. However, we miss any experimental evidence for this assumption.

3.5 CONCLUSION

In this chapter, we focussed on the construction of an alignment between process models – a prerequisite for any analysis of their behaviour consistency. We clarified the basic concepts of alignments and formally defined them for net systems. Further, we discussed the large body of work on model heterogeneity and related it to the setting of process models. A review of matching techniques for conceptual models in general revealed that there has been hardly any work on the identification of complex correspondences between model elements. For process models in particular, we are not aware of any work addressing this problem.

This points to a severe issue. Even models created of a single business process for the same purpose may show differences in the assumed abstraction level. Once process models are created

for different purposes, such conceptual heterogeneity is the rule not the exception. Hence, there is great demand for techniques that identify complex correspondences between process models. Despite this fact, the problem has been largely neglected in research.

The ICoP framework can be seen as a first step towards advancing the identification of correspondences between process models. It provides a modular and adaptable architecture for the implementation of matchers, by splitting up matchers into searcher, booster, evaluator and selector components. This allows for the development of matchers that detect complex 1:n correspondences. It can be extended towards complex n:m correspondences, even though this may require new heuristics to select groups of activities for analysis. The combinatorial problem is increased even further for this kind of correspondences.

Our experimental results highlight that we are able to identify a significant number of complex 1:n correspondences. Although this demonstrates the potential of our framework for improving matching results, we also explicated that, compared to existing 1:1 matchers, the increase in recall is often traded for a decrease in precision. Finally, a minimal level of textual similarity for similar activities is required for the matchers to produce acceptable results. In our experiments, this requirement was met if a standard process was matched with its implementations, but not for the case of models stemming from different organisations. Against the background of different modelling purposes, the extent to which correspondences may be discovered depends on the impact of the modelling purpose on the used terminology. For various modelling drivers, this impact can be assumed to be rather modest. For instance, models created for documentation and models created for automation are likely to refer to the same business objects as standardised by the terminology of an organisation. However, we can also think of exceptional cases. For instance, models created for certification may explicitly deviate from the terminology of an organisation and adopt the terminology of the certification body.

We conclude that the construction of alignments between process models is a conceptually challenging task. A full-fledged automatic identification of complex correspondences does not appear to be realistic. However, the ICoP framework shows that at least semi-automated support for the identification of complex correspondences can be achieved. As such, the framework forms the basis for future investigations in this area.

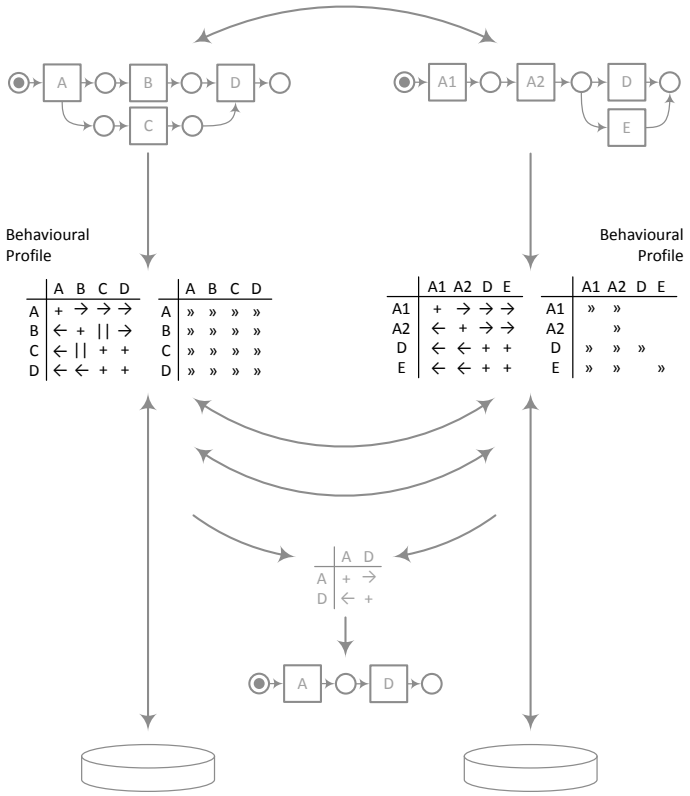
Part II

FOUNDATIONS OF BEHAVIOUR CONSISTENCY

4

BEHAVIOURAL PROFILES

This chapter is based on results published in [504, 505, 508].



BEHAVIOURAL profiles are an abstraction of the behaviour of a net system. Although the consistency analysis as proposed in this work is grounded on these profiles, they are a generic behavioural model and independent of a certain use case. This chapter is dedicated to the definition of different variants of behavioural profiles along with a discussion of their properties. Section 4.1 focusses on the notion of a behavioural profile. In Section 4.2, we extend this notion, which yields the causal behavioural profile. We lift both concepts to labelled net systems in Section 4.3. Section 4.4 is devoted to equivalences that ground in behavioural profiles. Section 4.5 reviews related behavioural concepts, before Section 4.6 concludes this chapter.

4.1 THE NOTION OF A BEHAVIOURAL PROFILE

This section presents basic definitions for behavioural profiles. Once the notion of a behavioural profile has been introduced, we elaborate on properties of such a profile.

Definitions

The behavioural profile captures behavioural characteristics of a net system under the assumption of trace semantics. Behavioural profiles capture the *order of potential occurrences* of transitions. They define order dependencies on the level of pairs of transitions. In particular, dependencies between *all* occurrences of two transitions are considered.

The definition of a behavioural profile is grounded on the notion of *weak order*. Informally, two transitions t_1 and t_2 are in weak order, if there exists a firing sequence starting in the initial marking in which t_1 occurs before t_2.

Definition 4.1.1 (Weak Order)
Let (N, M_0) be a net system with $N = (P, T, F)$ and $T' \subseteq T$ a set of transitions. A pair of transitions $(x, y) \in (T' \times T')$ is in the *weak order relation* $\succ$ over T', iff there exists a firing sequence $\sigma = \langle t_1, \ldots, t_n \rangle$ with $(N, M_0)[\sigma\rangle$ and indices $j, k \in \mathbb{N}$, $1 \leqslant j < k \leqslant n$, for which holds $t_j = x$ and $t_k = y$.

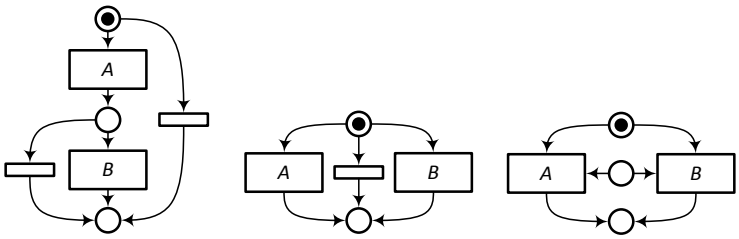
Depending on how two transitions of a system are related by weak order, we define three relations forming the behavioural profile. The behavioural profile is defined over a set of transitions of a net system.

Definition 4.1.2 (Behavioural Profile)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$ and $T' \subseteq T$ a set of transitions. A pair of transitions $(x, y) \in (T' \times T')$ can be in the following *profile relations*:

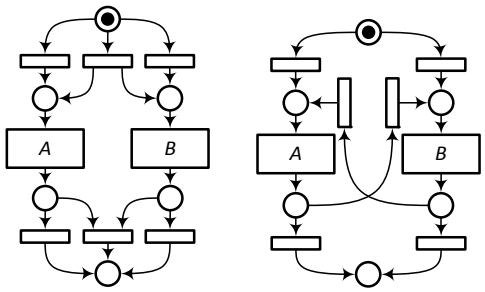
- The *strict order relation* $\rightsquigarrow$, iff $x \succ y$ and $y \not\succ x$.
- The *exclusiveness relation* $+$, iff $x \not\succ y$ and $y \not\succ x$.
- The *interleaving order relation* $\|$, iff $x \succ y$ and $y \succ x$.

$\mathcal{B} = \{\rightsquigarrow, +, \|\}$ is the *behavioural profile* over T'.

For a net system $S = (N, M_0)$ with $N = (P, T, F)$, we refer to the behavioural profile over T as the behavioural profile of S. The exemplary net systems in Figure 18 illustrate the relations of the behavioural profile for two dedicated transitions A and B. Figure 18a shows that the strict order relation enforces neither the occurrence of the first transition, nor of the second transition. For the two transitions in Figure 18a, it holds $A \rightsquigarrow B$. Hence, the strict order relation captures the fact that all occurrences of both transitions, A and B, are ordered in all firing sequences



(a) Strict order. (b) Exclusiveness. (c) Exclusiveness.



(d) Interleaving order. (e) Interleaving order.

Figure 18: Net systems that illustrate the relations of the behavioural profile for two transitions A and B.

reachable from the initial marking. In the system in Figure 18b, both transitions are exclusive to each other, $A + B$. That is, both transitions are never observed together in any firing sequence, and there may be a firing sequence that contains none of them. Even if the initial marking is dead, i. e., only the empty firing sequence is reachable from the initial marking, transitions are related according to the behavioural profile. In this case, the weak order relation is empty and all transitions are considered to be exclusive to each other. This case is illustrated in Figure 18c. Figure 18d and Figure 18e exemplify two reasons for interleaving order of two transitions, $A||B$. Potential concurrent enabling of two transitions or multiple interleaved firings of two transitions cause interleaving order between both transitions.

Properties

The introduced terminology for behavioural profiles is inspired by common notions known from order theory [101]. Still, none of the presented relations is a preorder, a partial order, or a total order, if the behaviour of the net system is not restricted. In particular, the strict order relation is not a strict partial order since it is not necessarily transitive for the generic class of net systems. Despite this fact, the definition of the profile relations implies certain properties. We observe that the relations are either antisymmetric and irreflexive, or symmetric.

Property 4.1.1. For any behavioural profile $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ over a set of transitions T' holds that the relation $\rightsquigarrow$ is antisymmetric and irreflexive, whereas relations $+$ and $\|$ are symmetric.

Let $\succ^{-1} = \{(y, x) \in T' \times T' \mid x \succ y\}$ the inverse relation for $\succ$. Then, this property follows immediately from the definition of strict order as $\rightsquigarrow = \succ \setminus \succ^{-1}$, exclusiveness as $+ = (T' \times T') \setminus (\succ \cup \succ^{-1})$, and interleaving order as $\| = \succ \cap \succ^{-1}$. This definition also shows that the profile relations are mutually exclusive. In other words, a pair of transitions is only in one of the behavioural relations.

Property 4.1.2. For any behavioural profile $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ over a set of transitions T' holds that the relations $\rightsquigarrow$, $+$, and $\|$ are mutually exclusive.

We further define the inverse relation of the strict order relation as $\rightsquigarrow^{-1} = \{(y, x) \in (T' \times T') \mid x \rightsquigarrow y\}$. We refer to $\rightsquigarrow^{-1}$ as the *reverse strict order* relation. Together with this relation, the profile relations show another important property. The four relations yield a partitioning of the Cartesian product of transitions over which they are defined.

Property 4.1.3. For any behavioural profile $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ over a set of transitions T' holds that the relations $\rightsquigarrow$, $\rightsquigarrow^{-1}$, $+$, and $\|$ partition $T' \times T'$.

Again, this property is verified by the definition of the profile relations based on the weak order relation. We earlier discussed the characterisation of the three profile relations using the weak order relation. Reverse strict order is defined as $\rightsquigarrow^{-1} = \succ^{-1} \setminus \succ$.

Cyclic net structures affect the profile relations, as illustrated by the example in Figure 18e. Here, transitions A and B are part of a circuit. This yields interleaving order not only for the transition pair (A, B), but also for the self-relations of both transitions. The identity relation over transitions is partitioned by interleaving order and exclusiveness.

Property 4.1.4. For any behavioural profile $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ over a set of transitions T' holds that a transition $t \in T'$ is either exclusive to itself, $t + t$, or in interleaving order to itself, $t \| t$.

To verify this property, we have to consider two cases, $(t, t) \in \succ$ and $(t, t) \notin \succ$. For the former, it holds that $(t, t) \in \succ$ implies $(t, t) \in \succ^{-1}$, which yields $t \| t$. For the latter, we have the inverse implication, i. e., $(t, t) \notin \succ$ implies $(t, t) \notin \succ^{-1}$ and, therefore, $t + t$. Consequently, the profile relations capture whether a transition may occur at most once, $t + t$, or whether it may occur multiple times in a firing sequence of the net system, $t \| t$.

Interleaving order as a self-relation may be caused by a circuit. However, it may also be due to potential self-concurrent enabling of the transition. For instance, it holds $A + A$ and $B \| B$ in Figure 19.

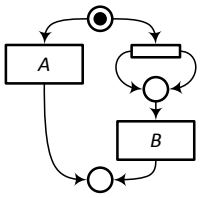


Figure 19: System with potential self-concurrent enabling.

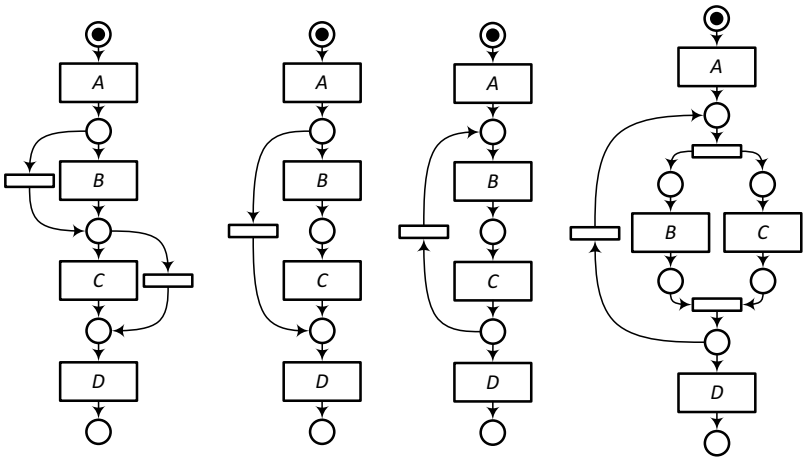
4.2 THE NOTION OF A CAUSAL BEHAVIOURAL PROFILE

After we introduced the notion of a behavioural profile, this section presents an extended variant of this behavioural abstraction. Behavioural profiles relate pairs of transitions according to their *order of potential occurrence*. A behavioural profile provides a rather coarse-grained behavioural abstraction. It is extended to take optionality and causality of transition occurrences into account.

Optionality of a transition is given, if there is a firing sequence starting in the initial marking that does not contain the transition and leads to a marking in which the transition is dead. Optionality can be lifted from single transitions to sets of transitions. A set of transitions is considered to be *jointly optional*, if all transitions are optional and any firing sequence starting in the initial marking that contains at least one of the transitions must not lead to a marking in which the remaining transitions are dead. As illustrated in Figure 20a and Figure 20b, this property cannot be derived from the knowledge about optionality of single transitions. In both systems, B and C are optional, but only in Figure 20b the set of transitions $\{B, C\}$ is jointly optional.

Closely related to optionality is *causality* of transition occurrences. This property requires that one transition can only occur after the occurrence of another transition. Hence, causality comprises two aspects, a certain *order* of occurrences and a *causal coupling* of occurrences. The former is addressed by the behavioural profile in terms of the strict order relation, whereas the latter is not captured. For instance, B is a cause of C in Figure 20b, but not in Figure 20a, even though both models show equal behavioural profiles.

We cope with causal dependencies between transition occurrences by extending the behavioural profile. The *causal* behavioural profile is a more fine-grained behavioural abstraction that provides a closer approximation of trace semantics. Technically, the causal behavioural profile introduces a co-occurrence relation for pairs of transitions. Two transitions are co-occurring, if any firing sequence starting in the initial marking that contains the first transition either contains the second transition or can be



(a) B and C are optional.

(b) B and C are jointly optional.

(c) Hidden causality for (B, C).

(d) Hidden causality for (B, C).

Figure 20: Net systems that illustrate the concepts of optionality and causality of transition occurrences.

continued such that it contains the second transition eventually. This relation captures only the joint occurrence, but does not impose any restriction on the order of occurrence within a firing sequence.

Definition 4.2.1 (Causal Behavioural Profile)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$, $T' \subseteq T$ a set of transitions, and $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ the behavioural profile over T'.

- A pair $(x, y) \in (T' \times T')$ is in the *co-occurrence relation* $\gg$, iff for all firing sequences σ with $(N, M_0)[\sigma\rangle(N, M')$ and $x \in \sigma$ it holds either $y \in \sigma$ or y is not dead in M'.
- $\mathcal{CB} = \{\rightsquigarrow, +, \|, \gg\}$ is the *causal behavioural profile* over T'.

For a net system $S = (N, M_0)$ with $N = (P, T, F)$, we refer to the causal behavioural profile over T as the causal behavioural profile of S. Trivially, the co-occurrence relation subsumes the identity relation over transitions. Hence, it holds $t \gg t$ for all transitions over which the causal behavioural profile is defined.

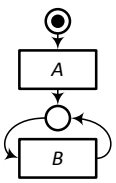


Figure 21: System with an L3-live transition, which can be fired infinitely often.

The definition of co-occurrence does not require any notion of final states of the net system. Hence, co-occurrence can be

decided even if a state (or a set of states) is never left. As an example, consider the net system in Figure 21. Here, transition B is L3-live [329], meaning that it appears infinitely often in some firing sequence starting in the initial marking. It holds $B \gg A$ as every firing sequence that contains transition B, contains transition A as well. It holds $A \gg B$, because transition B is not dead in the marking reached from the initial marking by firing A.

The co-occurrence relation of the causal behavioural profile allows for conclusions on optionality and causality. A single transition $t \in T$ of a net system $S = (N, M_0)$, $N = (P, T, F)$, is optional, if $t_0 \not\gg t$ for some transition t_0 enabled in the initial marking, $(N, M_0)[t_0\rangle$. A set $T' \subseteq T$ of transitions is optional, if all transitions themselves are optional and they are pairwise co-occurring to each other, $(T' \times T') \subseteq \gg$. A causal dependency between two transitions $t_1, t_2 \in T$ is observed, if they are in strict order, $t_1 \rightsquigarrow t_2$, and occurrence of the second implies occurrence of the first, $t_2 \gg t_1$.

In contrast to the behavioural profile, the causal behavioural profile differs for both systems in Figure 20a and Figure 20b. Further, we observe causality for transitions B and C in Figure 20b, but not in Figure 20a according to the relations of the causal behavioural profile. However, the systems in Figure 20c and Figure 20d show equal profiles despite their different trace semantics. The difference between these systems is not manifested in a dependency between *all* occurrences of a pair of transitions. Instead, the difference stems from different interleavings of transitions B and C, which is neglected by the relations of the behavioural profile. As a consequence, the interpretation of causality based on causal behavioural profiles differs from common definitions of causality. Those notions typically consider causal dependencies between *single* occurrences of transitions, e. g., the *response / leads-to* dependencies that are defined in temporal logic [342, 137, 452, 462]. Such a notion of causality between transitions B and C in Figure 20c and Figure 20d is hidden when using a behavioural abstraction such as causal behavioural profiles.

The co-occurrence relation of the causal behavioural profile is largely independent of the relations of the behavioural profile. The only two conclusions that can be drawn relate to the co-occurrence of dead transitions and exclusive transitions.

Property 4.2.1. For any transition holds, if it is dead in the initial marking, it is co-occurring with all other transitions of the net system.

If a transition is dead in the initial marking of the net system, it cannot be enabled in any reachable marking. Hence, there

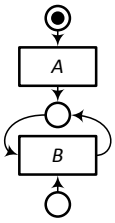


Figure 22: System with a dead transition.

is no firing sequence comprising the transition, so that the co-occurrence holds trivially regarding all other transitions.

We illustrate this property with the net system depicted in Figure 22. Here, transition B is dead. As there is a firing sequence starting in the initial marking and containing transition A, we arrive at $A \not\gg B$. However, it holds $B \gg A$. All firing sequences containing transition B, contain transition A as well or lead to a marking in which transition A is not dead. This requirement is trivially satisfied by the absence of any firing sequence containing transition B.

Property 4.2.2. For any pair of transitions holds, if they are not dead in the initial marking and exclusive according to the behavioural profile, they are not co-occurring in either direction.

Since both transitions are not dead, each transition can be part of at least one firing sequence that starts in the initial marking. Then, this property follows immediately from the definition of the respective relations. Exclusiveness requires the absence of a firing sequence that starts in the initial marking of a net system and comprises both transitions. Co-occurrence, in turn, requires that such a firing sequence that contains one of the transitions either contains the other transition or leads to a marking in which the other transition is not dead. A pair of transitions cannot satisfy both requirements.

The causal behavioural profile is an extension of the behavioural profile. In the remainder of this thesis, therefore, we use the term behavioural profile to refer to both concepts whenever the difference between the causal and non-causal variant is not of importance.

4.3 ON LABELLED SYSTEMS

Up to now, we discussed the concept of behavioural profiles solely for unlabelled net systems. Still, behavioural profiles can be lifted to labelled net systems to come to a behavioural abstraction over labels of a net system. In this section, we first introduce preliminaries for labelled systems. Then, we define the behavi-

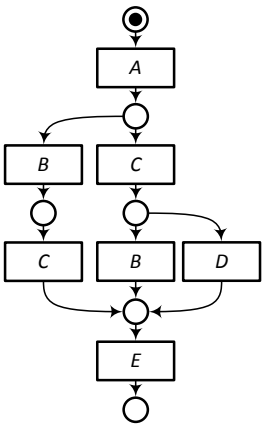


Figure 23: Labelled net system.

oural profile and the causal behavioural profile over the labels of a net system.

Preliminaries

A labelled net system comprises a net system and labelling information for all transitions.

Definition 4.3.1 (Labelled Net System)
A *labelled net* is a tuple $N = (P, T, F, \Lambda, \lambda)$, where (P, T, F) is a net, Λ is a set of labels, and $\lambda : T \to \Lambda$ is a labelling function. A system (N, M_0) is called *labelled net system*, if N is a labelled net.

Figure 23 depicts an example labelled net system. This net system comprises two pairs of transitions that carry equal labels, B and C, respectively. For a labelled net system (N, M_0) with $N = (P, T, F, \Lambda, \lambda)$, any firing sequence $\sigma = \langle t_1, \ldots, t_n \rangle$ is interpreted as a sequence of labels $\sigma_\lambda = \langle l_1, \ldots, l_n \rangle$ of the respective transitions, $\lambda(t_i) = l_i$ for all $1 \leqslant i \leqslant n$. As a short-hand notation, we write $l \in \sigma$ with $l \in \Lambda$ if there exists a transition $t \in T$ with $t \in \sigma$ and $\lambda(t) = l$.

Behavioural Profile on Labels

To define the behavioural profile on labels, we lift the weak order relation to labels as follows. Two labels are in weak order, if and only if two transitions carrying those labels are in weak order.

Definition 4.3.2 (Weak Order on Labels)
Let $S = (N, M_0)$ be a labelled system with $N = (P, T, F, \Lambda, \lambda)$, $\succ$ the weak order relation over T, and $\Lambda' \subseteq \Lambda$ a set of labels. A pair of labels $(l_1, l_2) \in (\Lambda' \times \Lambda')$ is in the *weak order relation on labels* $\succ_\Lambda$ over Λ', iff there are transitions $x, y \in T$ such that $\lambda(x) = l_1$, $\lambda(y) = l_2$, and $x \succ y$.

The relations of the behavioural profile are defined for labels as follows.

Definition 4.3.3 (Behavioural Profile on Labels)
Let $S = (N, M_0)$ be a labelled system with $N = (P, T, F, \Lambda, \lambda)$ and $\Lambda' \subseteq \Lambda$ a set of labels. A pair of labels $(l_1, l_2) \in (\Lambda' \times \Lambda')$ is in the following *profile relations on labels*:

- The *strict order relation* $\rightsquigarrow_{\Lambda'}$, if $l_1 \succ_{\Lambda'} l_2$ and $l_2 \not\succ_{\Lambda'} l_1$.
- The *exclusiveness relation* $+_{\Lambda'}$, if $l_1 \not\succ_{\Lambda'} l_2$ and $l_2 \not\succ_{\Lambda'} l_1$.
- The *interleaving order relation* $\|_{\Lambda'}$, if $l_1 \succ_{\Lambda'} l_2$ and $l_2 \succ_{\Lambda'} l_1$.

The set $\mathcal{B}_{\Lambda'} = \{\rightsquigarrow_{\Lambda'}, +_{\Lambda'}, \|_{\Lambda'}\}$ is the *behavioural profile on labels* over Λ'.

We illustrate the relations of the behavioural profile on labels with the example depicted in Figure 23. A transition labelled with B is exclusive to a transition labelled with C. Once the behavioural profile is lifted from transitions to labels, however, both labels are in interleaving order. Even though there are multiple transitions carrying the label B, all of them are exclusive to the transition labelled with D. Hence, the labels B and D are exclusive according to the behavioural profile on labels.

The behavioural profile on labels is derived directly from the behavioural profile of a net system. Consequently, it is computed efficiently once the behavioural profile is known.

Proposition 4.3.1. *The following problem can be solved in* $O(n^2)$ *time with* n *as the number of transitions:*
Given the behavioural profile of a labelled net system, to derive its behavioural profile on labels.

Proof. Both, deriving the weak order relation on labels and setting the relations of the behavioural profile on labels, requires iteration over the Cartesian product of transitions of the net system. Assuming that each label relates to at least one transition, the number of labels is smaller or equal than the number of transitions. Thus, overall time complexity is $O(n^2)$ with n as the number of transitions. □

Causal Behavioural Profile on Labels

Similar to the behavioural profile, the co-occurrence relation of the causal behavioural profile is lifted to labels of transitions.

Definition 4.3.4 (Causal Behavioural Profile on Labels)
Let $S = (N, M_0)$ be a labelled system with $N = (P, T, F, \Lambda, \lambda)$, $\Lambda' \subseteq \Lambda$ a set of labels, and $\mathcal{B}_{\Lambda'} = \{\rightsquigarrow_{\Lambda'}, +_{\Lambda'}, \|_{\Lambda'}\}$ the behavioural profile on labels over Λ'.

- A pair of labels $(l_1, l_2) \in (\Lambda' \times \Lambda')$ is in the *co-occurrence relation on labels* $\gg_{\Lambda'}$, iff for all firing sequences σ starting in

$M_0, (N, M_0)[\sigma\rangle(N, M')$, with $l_1 \in \sigma$ it holds either $l_2 \in \sigma$ or there is a transition $t \in T$ such that $\lambda(t) = l_2$ and t is not dead in M'.

- The set $\mathcal{CB}_{\Lambda'} = \{\rightsquigarrow_{\Lambda'}, +_{\Lambda'}, \|_{\Lambda'}, \gg_{\Lambda'}\}$ is the *causal behavioural profile on labels* over Λ'.

For the example shown in Figure 23, for instance, co-occurrence is observed from label B to label C, but not vice versa. Any firing sequence starting in the initial marking that contains a transition labelled with B, either contains a transition labelled with C or can be continued with a transition labelled with C. The opposite, $C \gg B$, does not hold true.

In contrast to the behavioural profile, the causal behavioural profile on labels cannot be derived from the causal behavioural profile directly. Co-occurrence of labels cannot be deduced from co-occurrence of transitions when considering solely pairs of transitions. Consider labels A and C in the net system in Figure 23. Both labels are co-occurring according to Definition 4.3.4 as every firing sequence starting in the initial marking that contains the label A contains the label C or can be continued with label C. However, there is no co-occurrence between the transition labelled with A and those that are labelled with C.

After we elaborated on the relation of behavioural profiles and their counterparts on labels, we restrict the discussion to profiles for unlabelled net systems in the remainder of this thesis. According to Proposition 4.3.1, however, all computations that utilise non-causal behavioural profiles and are done for unlabelled systems can be lifted to labelled systems with a minor computational overhead.

4.4 BEHAVIOURAL PROFILE EQUIVALENCES

The abstraction of behavioural profiles may be utilised to compare the behaviour of two net systems. This section introduces equivalences based on behavioural profiles. Later, we discuss their relation to common notions of behaviour equivalence.

Definitions

First, the notion of a behavioural profile that focusses on the order of potential occurrence gives rise to a definition of equivalence. Two net systems are behavioural profile equivalent, if the relations of their behavioural profiles coincide with each other. We define this equivalence under the assumption of equal sets of transitions of two systems. Still, the definition may be lifted to any isomorphism between the transitions of two net systems.

Definition 4.4.1 (Behavioural Profile Equivalence)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems and $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2\}$ their behavioural profiles. S_1 and S_2 are *behavioural profile equivalent* denoted by $S_1 \equiv S_2$, iff $\rightsquigarrow_1 = \rightsquigarrow_2$, $+_1 = +_2$, and $\|_1 = \|_2$.

In the same vein, an equivalence notion is grounded on the notion of a causal behavioural profile. It requires all relations of the causal behavioural profile of two net systems to coincide with each other.

Definition 4.4.2 (Causal Behavioural Profile Equivalence)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems and $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1, \gg_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2, \gg_2\}$ their behavioural profiles. S_1 and S_2 are *causal behavioural profile equivalent* denoted by $S_1 \equiv_C S_2$, iff they are behavioural profile equivalent and $\gg_1 = \gg_2$.

For both equivalence definitions, the characteristic properties of an equivalence notion are indeed satisfied. That is, the presented relations are reflexive, symmetric, and transitive.

Theorem 4.4.1. *The relations $\equiv$ and $\equiv_C$ are equivalences.*

Proof. Both relations are reflexive, symmetric, and transitive.

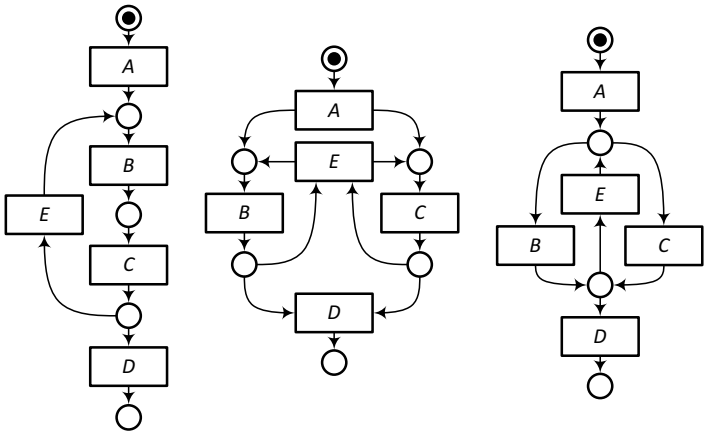
Reflexivity. For any net system S it holds $S \equiv S$ and $S \equiv_C S$ as the relations of the (causal) behavioural profile are uniquely derived from the existence of certain firing sequences.

Symmetry. Let S_1 and S_2 be net systems. Then, $S_1 \equiv S_2$ implies $S_2 \equiv S_1$ and $S_1 \equiv_C S_2$ implies $S_1 \equiv_C S_2$. This follows from the symmetry of set equivalence for the relations of the (causal) behavioural profile in Definition 4.4.1 and Definition 4.4.2.

Transitivity. Let S_1, S_2, and S_3 be net systems. Then, $S_1 \equiv S_2$ and $S_2 \equiv S_3$ implies $S_1 \equiv S_3$, and $S_1 \equiv_C S_2$ and $S_2 \equiv_C S_3$ implies $S_1 \equiv_C S_3$. This follows directly from the transitivity of set equivalence for the relations of the (causal) behavioural profile in Definition 4.4.1 and Definition 4.4.2.

□

Figure 24 illustrates equivalences based on (causal) behavioural profiles. All three net systems show equal behavioural profiles. The order of potential occurrence is equal for all pairs of transitions. For instance, transitions A and E are in strict order Transitions C and E are in interleaving order. Even though the three systems show differences in terms of trace semantics related to the circuits, i. e., transitions B, C, and E, the order dependencies between all occurrences of two transitions are equal in all three systems. In contrast, these differences partially affect the causal



(a) (b) (c)

Figure 24: All three systems are behavioural profile equivalent; (a) and (b) are causal behavioural profile equivalent.

behavioural profile. Only the systems in Figure 24a and Figure 24b are causal behavioural profile equivalent. Both systems show co-occurrence for transitions B and C in both directions. In the system in Figure 24c, both transitions are not co-occurring. As an example, a firing sequence consisting of transitions A, B, and D leads to a marking in which transition C is dead even though it has not been fired. Hence, it holds $B \not\gg C$ for the system in Figure 24c.

Behaviour Equivalences

Behavioural profile equivalences relate to common notions of behaviour equivalence that are grounded on different semantics for behavioural models. The seminal work of van Glabbeek [474, 476] classifies these semantics in the linear time – branching time spectrum, either for concrete sequential behavioural models [474] or for sequential behavioural models with silent transitions [476]. We focus on the former, as behavioural profiles are grounded on the observable behaviour of net systems.

The linear time – branching time spectrum for sequential behavioural models defines 11 semantics that yield 11 notions of behaviour equivalence. Branching bisimulation is seen as the upper bound of this spectrum, which requires that two behavioural models can simulate one another [478]. Hence, not only the observable behaviour, firing sequences in net systems, but also the moment of choice is taken into account. The lower bound of this spectrum is trace equivalence as introduced by Hoare [213]. Applied to net systems, it requires equivalence of the sets of firing sequences. Besides these two poles, various equivalences have

been proposed [474] and advocated to be applied in the field of business process analysis [209].

All the aforementioned equivalences assume sequential behavioural models, also referred to as interleaving semantics. Equivalences that are based on non-sequential semantics [354, 177, 50] are typically neglected for the comparison of business processes as all transitions may be split up into two transitions indicating the start and the end of the transition, respectively. For models extended in this way, equivalences for sequential behavioural models are applied, cf., [234]. A survey of equivalences for net systems under sequential and non-sequential semantics can be found in [363].

Behavioural profiles are an abstraction of trace semantics. This already suggests that equivalence of (causal) behavioural profiles is a weaker criterion for the comparison of two net systems than trace equivalence. The two net systems depicted in Figure 24a and Figure 24b illustrate that equivalence of causal behavioural profiles does not coincide with trace equivalence. For instance, the system in Figure 24b allows for the trace $\sigma = \langle A, C \rangle$, which is not possible in the system in Figure 24a. Nevertheless, we observe that trace equivalence implies the equivalence of causal behavioural profiles and, therefore, of behavioural profiles.

Proposition 4.4.2. *Every two net systems that are trace equivalent are also causal behavioural profile equivalent.*

Proof. If two net systems have the same set of traces, they end up with equal weak order relations as those are built on the existence of a certain trace. Both systems have equal co-occurrence relations, as co-occurrence of transitions is decided by exploiting all possible continuations for a certain trace. Hence, both systems are causal behavioural profile equivalent. □

4.5 RELATED BEHAVIOURAL CONCEPTS

The abstraction of behavioural profiles is related to other behavioural concepts. First, this section reviews approaches to relational semantics for the creation and analysis of behavioural models. Second, we focus on behavioural relations that have been proposed in process mining. This section closes with a discussion of behavioural profiles in the light of other approaches to behavioural abstraction.

Relational Semantics

Relational semantics have been proposed to reason on the consistency of hardware specifications [393]. The control logic cir-

cuit of such a specification may be represented by a labelled net system, in which each transition refers to a state change of a binary variable. The authors of [393] classify transitions of a net system to be sequential or parallel. Two transitions t_1 and t_2 are sequential, if t_1 precedes t_2 in all traces reachable from the initial marking. A formal definition of the assumed notion of precedence is not presented, even though an example suggests an interpretation for single transition occurrences. Hence, two transitions that may be fired multiple times in alternating order as part of a circuit would be considered to be sequential by the notion of [393], but in interleaving order according to the behavioural profile. In [393], the sequential and parallel relations along with an exclusiveness relation are also defined for operations of a programming language. This language is block-structured and limited to acyclic programs. Consequently, the relations are directly assigned to common language constructs, e. g., if-then-else constructs. Besides these differences, the ideas in [393] can be seen as the conceptual roots of behavioural profiles.

Behavioural relations are used for matching service descriptions in [144]. The authors define sequential, choice, and parallel relations for activities of BPEL processes. These relations are directly assigned to control flow routing constructs of BPEL processes. To take potential repetition of activities into account, all three relations may have multiplicity annotations. With these notions, different types of services matches are introduced. Behavioural profiles can be seen as a generalisation of these relations. They are defined for a generic behavioural model, whereas the relations in [144] are restricted by the underlying behavioural model. On the one hand, this model simplifies control flow dependencies by neglecting BPEL transition conditions and join conditions. On the other hand, BPEL processes do not support the definition of arbitrary cycles, cf., [456].

The notion of an *order matrix* has been introduced in the context of managing process variants [271, 272, 273, 274]. For pairs of activities, this matrix captures their behavioural dependencies by predecessor, successor, AND-block, XOR-block, and loop relations. These relations partition the Cartesian product of activities of a process model, apart from the self-relations. When neglecting the latter, these relations virtually coincide with the relations of the behavioural profile. Predecessors and successors are captured by strict order, XOR-blocks correspond to exclusiveness and interleaving order combines the AND-block and loop relation. Apart from this difference, the behavioural profile can be seen as a generalisation of the order matrix, which is introduced only for block-structured process models.

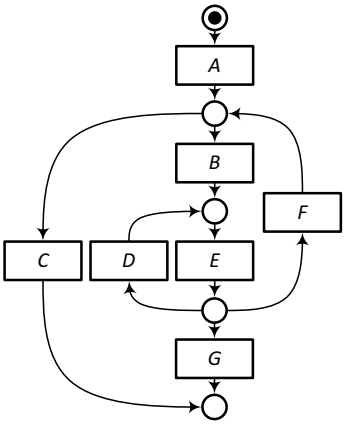
Capturing the behaviour of a system by a set of relations is also at the core of declarative approaches to process model-

ling. These approaches specify a set of constraints, typically grounded on Linear Temporal Logic [298], which restricts the possible behaviour of a system. Most prominently, the DecSerFlow language [452, 462, 327] defines a large set of behavioural constraints, such as mutual exclusiveness, precedence, and response dependencies. These constraints are more fine-grained than the relations of the behavioural profile and allow for the definition of dependencies between single occurrences of transitions. The constraints imposed by the relations of the behavioural profile, therefore, may be encoded in DecSerFlow. The idea to employ enabling and disabling constraints to define a behavioural model was also incorporated in the AMBER language [140]. Finally, relational semantics have been advocated for the specification of choreography models that focus on the interactions between different behavioural models on a global level [112]. The Let's Dance language [532, 531, 113] is based on behavioural relations such as precedes or inhibits dependencies that are defined between interactions.

Behavioural Relations in Process Mining

Relations similar to those of the behavioural profile are applied in process mining, which aims at the construction of process models from event logs, i. e., observed execution sequences [99, 11, 457, 460, 446]. Early work on process mining extracts dependency graphs from an event log [99, 11]. In [99], such a graph is derived by investigating the direct successorship of activity execution. Then, a probabilistic strategy is used to identify the most probable execution dependencies in the graph. Other work builds a dependency graph from an indirect follows-relation between pairs of activity executions [11]. This relation holds between two activities, if the first terminates before the second is started in all observed execution sequences. Hence, this relation resembles the weak order relation from which we derive the behavioural profile.

The α-algorithm aims at the construction of a WF-system from sequences of observed transition occurrences [457, 446]. It follows the idea of exploiting direct successorship of transition occurrences. A *directly follows* relation contains all transitions that succeed each other without any other transition occurring in between. Based on this relation, the α-algorithm defines three relations, $\rightarrow$, #, and $\|$. The three relations stem from the different combinations of the directly follows relation for pairs of transitions. Relation # holds between *'pairs of transitions that never follow each other directly'* [457]. A causal dependency $\rightarrow$ holds if a transition follows another transition in some firing sequence, but not vice versa. The relation $\|$ captures transition pairs for



(a)

(M1) Footprint α-Algorithm 1-Look-Ahead

	A	B	C	D	E	F	G
A	#	→	→	#	#	#	#
B	←	#	#	#	→	←	#
C	←	#	#	#	#	←	#
D	#	#	#	#	||	#	#
E	#	←	#	||	#	→	→
F	#	→	→	#	←	#	#
G	#	#	#	#	←	#	#

(M2) Footprint 2-Look-Ahead

	A	B	C	D	E	F	G
A	#	→	→	#	→	#	#
B	←	#	#	→	||	||	→
C	←	#	#	#	←	←	#
D	#	←	#	||	||	→	→
E	←	||	→	||	||	||	→
F	#	||	→	←	||	#	#
G	#	←	#	←	←	#	#

(M3) Behavioural Profile Far-Look-Ahead

	A	B	C	D	E	F	G
A	+	→	→	→	→	→	→
B	←	||	→	||	||	||	→
C	←	←	+	←	←	←	+
D	←	||	→	||	||	||	→
E	←	||	→	||	||	||	→
F	←	||	→	||	||	||	→
G	←	←	+	←	←	←	+

(b)

Figure 25: Overview of different relational semantics; (a) depicts an example net system, (b) shows three different relational semantics for this net system.

which the first follows the second, and vice versa. This is similar to the profile relations, which are deduced from different combinations of the weak order relation. Recently, these relations have been called *footprint* [446]. A footprint shows similar properties as the behavioural profile. The relations are mutually exclusive and partition the Cartesian product of transitions. Yet, the relations are different. The underlying directly follows relation emphasises direct causal dependencies, whereas the profile relations focus on indirect dependencies. Actually, there is a whole spectrum of relational semantics between the relations of the footprint and those of the behavioural profile, see Figure 25. For the firing sequences allowed for by the net system depicted in Figure 25a, the matrix (M1) depicts the relations of the α-algorithm. These relations implement a look-ahead of one when evaluating whether there exists an order dependency. As an example, it holds B # D as both transitions never occur together. Conceptually, we may increase the look-ahead when deriving the relations. Matrix (M2) in Figure 25a shows relations that are grounded on a follows relation that contains all pairs of transitions that follow each other directly or with just a single transition in between. Hence, it implements a look-ahead of two. Then, we observe an order dependency for the aforementioned transitions, B $\rightarrow$ D, as there exist a firing sequence in which B is followed by first E and then D. However, we do not observe any

sequence in which D is followed by B directly or with just one transition occurring in between (two transitions, E and F, have to occur in between). Following this line, the relations of the behavioural profile implement a far-look-ahead. Even though transitivity of the strict order relation holds solely for behavioural profiles of a certain class of net systems, the relations of the behavioural profile may be thought of the transitive closure of the order dependencies. Taking up the example again, we observe interleaving order for both transitions, B||D, as an occurrence of B may be followed by an occurrence of D, and vice versa. Information on ordering in cyclic structures is lost, whereas we obtain order dependencies that are independent of any transition occurrences that may happen in between. In contrast to the footprint, the behavioural profile captures exclusiveness for transitions do not occur together at all.

The look-ahead of one of the footprint also explains why the α-algorithm does not discover certain control flow constructs, such as loops of length one and two, and non-free-choice constructs. These constructs require increasing the look-ahead. Extended versions of the α-algorithm, the α^{+}-algorithm and the α^{++}-algorithm, incorporate respective relations. To cope with control flow loops of length two, the α^{+}-algorithm redefines the aforementioned footprint relations [104, 105]. By investigating not only direct successorship of transition occurrences but sequences of three transitions, multiple interleaving occurrences of two transitions and actual parallelism are distinguished. The α^{++}-algorithm aims at mining non-free-choice constructs and introduces further behavioural relations [512]. In particular, this algorithm leverages an indirect causality relation. It captures pairs of transitions that follow each other indirectly. This relation is close to our indirect strict order relation, yet different. It is sensitive to certain split and join patterns between the occurrences of two transitions, see [512]. The idea behind is to capture only indirect dependencies that are actually causalities, whereas our strict order relation also captures non-causal dependencies.

Further *matrix-based* representations of behaviour have been proposed for process mining. Genetic mining leverages the notion of a *causal matrix* [459, 106]. Such a matrix captures dependencies between net transitions by input and output condition functions. Those associate subsets of preceding or succeeding transitions to a single transition. Hence, they capture the combinations of transitions that must be fired to enable a certain transition, or can be fired after a certain transition has been fired. In contrast to our relations these relations assume a local perspective by focussing on direct predecessors and successors of a transition. A global perspective is assumed by the *follows* and *precedes* relations proposed to judge on the quality of mined pro-

cess models [395]. For a pair of transitions, these relations capture whether the first is never, sometimes, or always followed (preceded) by the second transition. There are various commonalities between these relations and behavioural profiles. In fact, the weak order relation underlying the behavioural profile corresponds to the follows relation with the values sometimes or always. Two transitions in weak order follow each other either in some or all firing sequences. As a consequence, the profile relations can be derived from the follows and precedes relations. For instance, two transitions that never follow and never precede each other are exclusive in the behavioural profile. Transitions that always follow or precede each other would by captured by the co-occurrence relation of the causal behavioural profile.

We summarise that many relations proposed in the context of process mining resemble the relations of the behavioural profile. Still, process mining typically focusses on causal dependencies, whereas the causal behavioural profile separates the order of potential occurrences and co-occurrence dependencies.

Behavioural Abstractions

The approaches to relational semantics discussed earlier are not intended to serve as a behavioural abstraction. In contrast, a behavioural profile provides an abstraction of trace semantics of a net system – a certain loss of behavioural detail is the desired outcome.

Approximation of trace semantics of net systems has been the motivation for the definition of causal footprints [470, 473]. Such a footprint captures semantics for a set of transitions by two relations, look-back links and look-ahead links. For a transition t, a look-back link defines a set of transitions of which one must have occurred sometime before t. A look-ahead link for a transition t defines a set of transitions, such that an occurrence of t is eventually followed by the occurrence of one of these transitions. A behavioural model is said to be consistent with a causal footprint, if all firing sequences reachable from the initial marking satisfy the constraints imposed by the footprint. Similar to behavioural profiles, causal footprints are a behavioural abstraction, so that net systems with different trace semantics may show equal causal footprints. However, unlike the notion of behavioural profiles, there are also various causal footprints for a single net system. It is suggested to address this issue by computing the closure of causal footprints to obtain a *'more informative footprint'* [470]. Still, this computation requires soundness of the underlying behavioural model and, therefore, is not applicable in the general case.

A communication fingerprint is a behavioural abstraction that focusses on potential interactions of a behavioural model [435, 337]. The concept has been defined for open net systems, i. e., net systems that are extended by means for synchronous and asynchronous communication. The protocol induced by the net system's behaviour is abstracted by a communication fingerprint as a set of constraints on message exchanges. In particular, message occurrence counts are defined that impose boundaries and dependencies regarding the cardinalities of consumed or produced messages. Communication fingerprints abstract from any control flow dependencies and focus on the interactions. Hence, this work is orthogonal to our work on behavioural profiles.

4.6 CONCLUSION

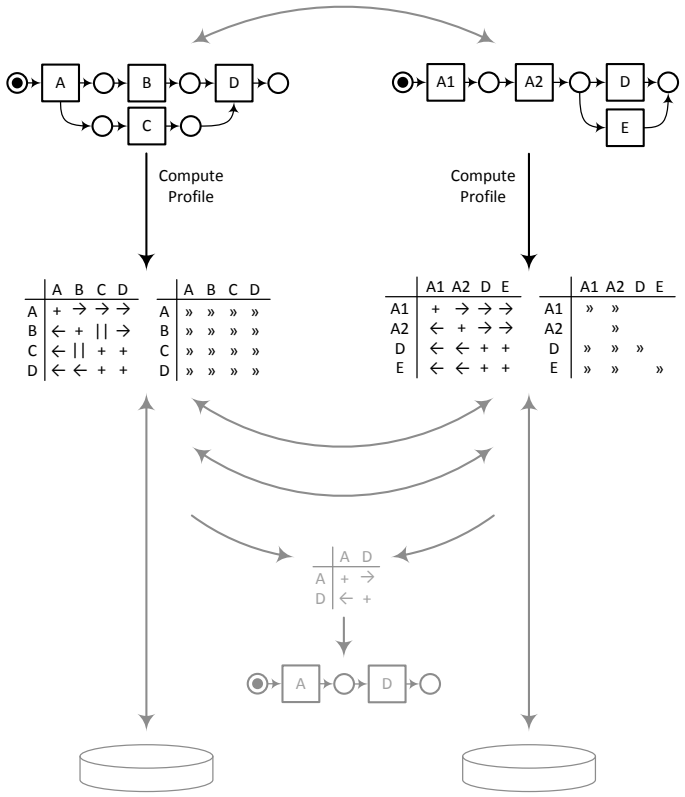
This chapter introduced the notion of behavioural profiles. Behavioural profiles capture behavioural characteristics of a net system by relations between pairs of transitions. We introduced two variants of behavioural profiles. The (non-causal) behavioural profile focusses on the order of potential occurrences of transitions. The causal behavioural profile extends the behavioural profile by a relation that captures co-occurrence dependencies.

We showed that both notions are the basis for the definition of equivalences. Behavioural profiles are an abstraction of the trace semantics of a net system. Therefore, these equivalences are weaker than trace equivalence.

5

COMPUTATIONS OF BEHAVIOURAL PROFILES

This chapter is based on results published in [504, 505, 508, 502].



We introduced behavioural profiles as an abstraction of the behaviour of a net system. The relations forming the behavioural profile follow from the existence of certain firing sequences. Hence, construction of a behavioural profile requires the analysis of all firing sequences or of all paths of the state space, respectively. This is known to require exponential space and time for arbitrary net systems [279, 444]. For dedicated classes of net systems, however, computation of behavioural profiles is efficient – requires low polynomial time.

In this chapter, we introduce techniques for the efficient computation of behavioural profiles. In Section 5.1, we focus on sound free-choice WF-systems. Systems in this class are known to show a close relation between structure and semantics. We leverage these results to derive behavioural profiles from the net structure. Section 5.2 complements these results by showing how structural decomposition of sound free-choice WF-systems

is leveraged to support partial computation of behavioural profiles. Net systems that do not meet the assumption of soundness and free-choiceness are addressed in Section 5.3. We introduce the computation of behavioural profiles for a bounded net system from its complete prefix unfolding. The latter refers to a specific representation of the state space of the net system. Although this technique is computationally hard, the underlying formalism explicitly aims at representing the system's behaviour in a compact way. Then, we investigate the applicability of our techniques in a real-world setting. Section 5.4 presents an implementation of all techniques for the computation of behavioural profiles and reports on findings from experiments with model collections from industry. Section 5.5 reviews work related to the presented computation techniques. Finally, Section 5.6 concludes this chapter.

5.1 COMPUTATIONS FOR SOUND FREE-CHOICE WF-SYSTEMS

For the class of *sound free-choice* WF-systems, we derive the behavioural profile directly from the net structure. Although we introduced the soundness and free-choice property in Section 2.2, the implications of these properties deserve further explanation. Both properties together imply a tight coupling of syntax and semantics of net systems [448, 234]. To discuss this coupling, we need the notion of a home marking. For a net system $S = (N, M_0)$, a marking M is a home marking of S, if it is reachable from every marking reachable from the initial marking, i. e., $M \in [N, M'\rangle$ for all markings $M' \in [N, M_0\rangle$.

In a live and bounded free-choice net system (N, M_0), the existence of a path from a place q to a place p, with M_p being a home marking, implies the existence of a firing sequence that consists of all transitions on the path between q and p, cf., Lemma 4.2 in [234]. Soundness of a WF-system (N, M_i) implies liveness of its short-circuit system (N', M_i) [448]. All markings reachable from the initial marking M_i in N are home markings in (N', M_i). Consequently, the result on the relation between the existence of paths and according firing sequences is applicable for paths in sound free-choice WF-systems. Further, our computations leverage that any sound free-choice WF-system is known to be safe, cf., Lemma 1 in [448].

The soundness and free-choiceness properties are verified in polynomial time. The free-choice property is decided based on the structure of the net system, i. e., the flow relation. Soundness of a net system is traced back to liveness and boundedness of the short-circuit system. It is decided in polynomial time for free-choice WF-systems [447]. Hence, deciding whether a given net

system is a sound free-choice WF-system is done in polynomial time to its size.

In the remainder of this section, first, we derive the relations of the behavioural profile from the structure of a sound free-choice WF-system. Second, we elaborate on the derivation of the co-occurrence relation for sound free-choice WF-systems that are S-systems, T-systems, or acyclic systems.

Derivation of the Profile Relations

To derive the relations of the behavioural profile, we need an auxiliary relation that captures concurrent enabling of transitions. To this end, we rely on the concurrency relation as introduced in [246, 247].

Definition 5.1.1 (Concurrency Relation)
Let (N, M_0) be a net system with $N = (P, T, F)$. The *concurrency relation* $\|_{co} \subseteq T \times T$ contains all transition pairs (x, y), such that there is a marking $M \in [N, M_0\rangle$ that enables them concurrently, i. e., $M \geqslant M_x + M_y$.

Transitions that show interleaving order according to the behavioural profile are not necessarily enabled concurrently in a marking reachable from the initial marking. Still, there is the following dependency between interleaving order and the concurrency relation.

Lemma 5.1.1. *For any free-choice system holds, every pair of transitions that is concurrent is also in interleaving order.*

Proof. Let (N, M_0) be a free-choice net system with $N = (P, T, F)$, $x, y \in T$, and $x \|_{co} y$. From the latter, we know that there is a marking $M \in [N, M_i\rangle$ with $M \geqslant M_x + M_y$. There are two firing sequences $(N, M)[xy\rangle$ and $(N, M)[yx\rangle$, which yields $x \succ y$ and $y \succ x$, i. e., interleaving order. □

With this result, we relate structural relations between transitions in sound free-choice WF-systems to the profile relations. For a sound free-choice WF-system (N, M_i) with $N = (P, T, F)$, we say that two transitions $x, y \in T$ are *cyclic dependent*, if $x\ F^+\ y$ and $y\ F^+\ x$. They are *structurally ordered*, if $x\ F^+\ y$ and $y\ \not{F^+}\ x$, and *structurally exclusive*, if $x\ \not{F^+}\ y$ and $y\ \not{F^+}\ x$. Each of these structural relations is close to one of the profile relations.

Lemma 5.1.2. *For any sound free-choice WF-system holds, for every two transitions that are not concurrent, interleaving order coincides with cyclic dependency.*

Proof. Let (N, M_i) be a sound free-choice WF-system with $N = (P, T, F)$, $x, y \in T$, and $x \not\|_{co} y$.

⇒ Let $x \| y$ and assume $x \not\mathrel{F^+} y$. Since it holds $x \| y$, there is a firing sequence containing x before y. Let $M_1, M_2 \in [N, M_i\rangle$ be the markings before and after firing of x as part of this firing sequence, i. e., $(N, M_1)[x\rangle(N, M_2)$. Due to $x \not\mathrel{F^+} y$, all places $x^\bullet$ cannot affect the enabling of y. This, along with soundness of the net system, implies that there is a marking $M_3 \in [N, M_2\rangle$ enabling y for which holds $M_3 \geqslant M_y + \sum_{p \in x^\bullet}(M_p)$. In other words, M_3 is reachable from M_2, marks all places $x^\bullet$, and enables y. Then, there must also be a marking M_4 from which M_3 is derived by firing of transition x, $(N, M_4)[x\rangle(N, M_3)$. Hence, $M_4 \geqslant M_x + M_y$, a contradiction with $x \not\|_{co} y$. Following the same argument, the assumption of $y \not\mathrel{F^+} x$ also results in a contradiction.

⇐ From $x \mathrel{F^+} y$ and $y \mathrel{F^+} x$ we know that, due to the soundness and the free-choice property, there must be a firing sequence containing both transitions in either order [234]. Hence, it holds $x \succ y$ and $y \succ x$, which yields interleaving order.

□

Structural exclusiveness is also related to the profile relations for sound free-choice WF-systems.

Lemma 5.1.3. *For any sound free-choice WF-system holds, for every two transitions that are not concurrent, exclusiveness coincides with structural exclusiveness.*

Proof. Let (N, M_i) be a sound free-choice WF-system with $N = (P, T, F)$, $x, y \in T$, and $x \not\|_{co} y$.

⇒ Let $x + y$ and assume $x \mathrel{F^+} y$. Since $x + y$, we know $x \not\succ y$. As the system is sound, x must not be a dead transition. Let $M_1, M_2 \in [N, M_i\rangle$ be the markings before and after firing of x in a firing sequence, $(N, M_1)[x\rangle(N, M_2)$. $x \mathrel{F^+} y$ implies a firing sequence containing x and y due to the soundness and the free-choice property [234]. There is a marking $M_3 \in [N, M_2\rangle$ with $M_3 \geqslant M_y$, i. e., y is enabled. This yields a contradiction with $x \not\succ y$. The argument is followed in reverse direction for the assumption of $y \mathrel{F^+} x$.

⇐ Let $x \not\mathrel{F^+} y$ and assume $x \succ y$. x must not be a dead transition (soundness property). Thus, it is contained in a firing sequence and there are two markings $M_1, M_2 \in [N, M_i\rangle$ before and after firing of x. To meet $x \succ y$, still, there has to be a marking $M_3 \in [N, M_2\rangle$ that enables y. As before , $x \not\mathrel{F^+} y$ implies that all places $x^\bullet$ cannot affect the enabling of y. Thus, the marking M_3 may enable y, while marking all places $x^\bullet$, $M_3 \geqslant M_y + \sum_{p \in x^\bullet}(M_p)$. Consequently, there is a marking M_4 from which M_3 is derived via firing of x, $(N, M_4)[x\rangle(N, M_3)$. Then, $M_4 \geqslant M_x + M_y$, which is not in line with $x \not\|_{co} y$. The argument is followed in reverse direction for the assumption of $y \succ x$. Thus, we conclude $x + y$.

□

As a next step, structural order of transitions is related to the profile relations for sound free-choice WF-systems.

Lemma 5.1.4. *For any sound free-choice WF-system holds, for every two transitions that are not concurrent, strict order coincides with structural order.*

Proof. Let (N, M_i) be a sound free-choice WF-system with $N = (P, T, F)$, $x, y \in T$, and $x \not\|_{co} y$. Both directions of the Lemma follow directly from Lemma 5.1.2 and Lemma 5.1.3, as both transitions x and y must not be structurally exclusive or cyclic dependent. Hence, strict order $x \rightsquigarrow y$ coincides with either $x\ F^+\ y$ and $y\ \not{F^+}\ x$, or $x\ \not{F^+}\ y$ and $y\ F^+\ x$. The latter is not possible as $y\ F^+\ x$ would imply $y \succ x$ due to soundness and free-choiceness [234]. □

Finally, we are able to state that the profile relations are derived from the concurrency relation and the flow relation of a sound free-choice net system.

Theorem 5.1.5. *For a sound free-choice WF-system, the behavioural profile is computed from its concurrency relation and its flow relation.*

Proof. Interleaving order is traced back to concurrency and cyclic dependencies by Lemma 5.1.1 and Lemma 5.1.2. Exclusiveness is derived from the flow relation using Lemma 5.1.3. Strict order is derived according to Lemma 5.1.4. As the relations of the behavioural profile partition the Cartesian product of transitions, cf., Property 4.1.3, this yields the behavioural profile for a sound free-choice WF-system. □

Even though the concurrency relation captures behavioural aspects of a system, it is computed in low polynomial time for the investigated class of systems. This enables efficient computation of a behavioural profile for a sound free-choice WF-system.

Corollary 5.1.6. *The following problem can be solved in $O(n^3)$ time with n as the number of transitions and places of the system:*
For a sound free-choice WF-system, to compute its behavioural profile.

Proof. Computation of the concurrency relation is done in $O(n^3)$ time for any free-choice system with n as the number of transitions and places of the system [247, 147]. Computation of the transitive closure of a relation over a set with n elements takes $O(n^3)$ time [486]. By Theorem 5.1.5, this suffices to derive the behavioural profile. □

We illustrate the presented results with the sound free-choice WF-system depicted in Figure 26. Transitions C and D are in

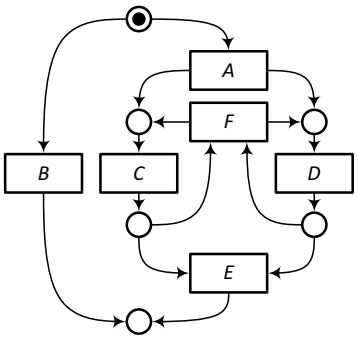


Figure 26: Example WF-system that is sound and free-choice.

the concurrency relation. Hence, we conclude interleaving order as their profile relation. Transitions A and C are structurally ordered, transitions C and F are cyclic dependent, and transitions A and B are structurally exclusive. All these pairs are not concurrent. Therefore, we derive strict order for transitions A and C, interleaving order for transitions C and F, and exclusiveness for transitions A and B.

Derivation of the Co-occurrence Relation

Having addressed the relations of the behavioural profile, we turn the focus on the co-occurrence relation of the causal behavioural profile. To characterise co-occurring transitions we first need an auxiliary result on the relation between co-occurrence and conflict-free paths of a sound net system. As usual, given a WF-net $N = (P, T, F)$, a path $\pi_N(x_1, x_k)$ is *forwards conflict-free*, if and only if $x_i \in (P \cap \pi_N\{x_1, x_k\})$ implies $|x_i\bullet| = 1$ for $1 \leqslant i < k$. The path $\pi_N(x_1, x_k)$ is *backwards conflict-free*, if and only if $x_i \in (P \cap \pi_N\{x_1, x_k\})$ implies $|\bullet x_i| = 1$ for $1 < i \leqslant k$.

Lemma 5.1.7. *For any transitions x and y in a sound WF-system holds,*

- *if there is a forwards conflict-free path from x to y, then $x \gg y$.*
- *if there is a backwards conflict-free path from x to y, then $y \gg x$.*

Proof. Let (N, M_i) be a sound WF-system with $N = (P, T, F)$ and $x, y \in T$.

- Consider the case $y \in (x\bullet)\bullet$. Then, for all places $p \in x\bullet$ on the path from x to y holds $|p\bullet| = 1$. Therefore, $p\bullet = \{y\}$. Then, every firing sequence containing x either leads to a marking that marks place p, or contains y as well. In the former case, y is not dead as soundness guarantees the reachability of M_o from any marking reachable from M_i. Hence, it holds $x \gg y$. Consider the case $y \notin (x\bullet)\bullet$ and let $t \in T$ be a transition with $x\ F^+\ t$ and $t\ F^+\ y$. For all places $p \in \bullet t$ holds $|p\bullet| = 1$, so that $p\bullet = \{t\}$. Consequently, for any

two markings $M_1, M_2 \in [N, M_i\rangle$ with $(N, M_1)[\sigma\rangle(N, M_2)$, $(N, M_1)[t\rangle$, and not $(N, M_2)[t\rangle$ it holds $t \in \sigma$. Starting with the transitions in $(x\bullet)\bullet$, all transitions in $\pi_N\{x, y\}$ have to be fired once they have been enabled to empty the one or more places of their pre-set. Again, soundness guarantees that M_o may be reached from all markings reachable from M_i that enable x. Consequently, any firing sequence containing x either contains y or leads to a marking in which y is not dead, i. e., $x \gg y$.

- The claim trivially holds by following the same argument in the reverse direction.

□

With this result, we characterise co-occurrence for transitions of sound workflow T-systems.

Theorem 5.1.8. *For any sound workflow T-system holds, all pairs of transitions are co-occurring.*

Proof. Let (N, M_i) be a sound workflow T-system with $N = (P, T, F)$. Let $i\bullet = \{t_i\}$ be the initial transition – there is only one because of the structure of T-systems. For any transition $t \in T$, any path $\pi_N(t_i, t)$ is forwards conflict-free, so that $t_i \gg t$ by Lemma 5.1.7. All firing sequences starting with t_i either contain transition $t \in T$ or lead to a marking in which t is not dead. Since every firing sequence starting in M_i contains t_i, all transitions are pairwise co-occurring. □

For sound workflow S-systems, we derive co-occurrence using the notions of dominators and post-dominators known from graph theory [13, 284]. Let $N = (P, T, F)$ be a WF-net with the initial place i and the final place o. For two nodes $x, y \in (T \cup P)$, x is a dominator of y, if and only if for all paths $\pi_N(i, y)$ it holds $x \in \pi_N\{i, y\}$. x is a post-dominator of y, if and only if for all paths $\pi_N(y, o)$ it holds $x \in \pi_N\{y, o\}$.

Theorem 5.1.9. *For any sound workflow S-system holds, two distinct transitions x and y are co-occurring, if and only if y is dominator or post-dominator of x.*

Proof. Let (N, M_i) be a sound workflow S-system with $N = (P, T, F)$ and $x, y \in T$. In a workflow S-system, every marking $M \in [N, M_i\rangle$ marks exactly one place, as only i is marked initially and for all transitions $t \in T$ we know $|\bullet t| = 1 = |t\bullet|$. Therefore, for every firing sequence $\sigma = \langle t_1, \ldots, t_n\rangle$ there is a path $\pi_N(t_1, t_n)$ containing all transitions of σ in the respective order.

- $\Rightarrow$ Let y be a dominator of x. Then, it holds $y \in \pi_N\{i, x\}$ for every path $\pi_N(i, x)$. Thus, any firing sequence σ with $(N, M_i)[\sigma\rangle(N, M_1)$ with $(N, M_1)[x\rangle$ is required to contain y, i. e., $x \gg y$. If y is a post-dominator of x, y is not dead in

all markings $M_2 \in [N, M_1\rangle$ since all paths $\pi_N(x, o)$ contain y and soundness guarantees that M_o is reachable from M_2.

$\Leftarrow$ Let $x \gg y$ and assume that y is neither a dominator nor a post-dominator of x. Since $x \gg y$, a firing sequence σ with $x \in \sigma$ and $(N, M_i)[\sigma\rangle(N, M_1)$ either contains y or y is not dead in M_1. If y is not a dominator, y is not necessarily part of σ. Then, any firing sequence starting in M_1 and ending in M_o (such a sequence exists by soundness) has to contain y. Hence, all paths $\pi_N(x, o)$ have to contain y, a contradiction with the assumption of y not being a post-dominator of x. □

Finally, we consider sound free-choice WF-systems that are acyclic. This class of systems does not subsume the two aforementioned classes. A sound workflow S-system may contain circuits.

In principle, we trace co-occurrence of transitions back to the exclusiveness relation. The main idea is described as follows. If there is no co-occurrence from one transition to another transition, a third transition is fired instead of the second transition in some marking. Informally, this third transition represents the *detour* that implements skipping of the second transition. In the absence of a circuit, the detour is manifested in the behavioural profile. The third transition is exclusive to the second transition, but not to the first one.

In a sound WF-system, there are no dead transitions. Hence, two transitions that are exclusive to each other cannot be co-occurring, cf., Property 4.2.2. Therefore, we neglect this case in the following statement.

Theorem 5.1.10. *For any sound free-choice WF-system holds, two distinct transitions x and y that are not exclusive, $x \not\# y$, and y is not part of a circuit, $y \not F^+ y$, are co-occurring, if and only if all transitions exclusive to y are exclusive to x.*

Proof. Let (N, M_i) be a sound free-choice WF-system with $N = (P, T, F)$, $x, y \in T$, $x \not\# y$, and $y \not F^+ y$. In this proof, we use the results obtained in Lemmas 5.1.1 to 5.1.4 without referring to the lemmas explicitly.

$\Leftarrow$ Let $(t + y) \Rightarrow (t + x)$ for all transitions $t \in T$ and assume $x \not\gg y$. The relations of the behavioural profile partition the set $T \times T$. As it holds $x \not\# y$, we distinguish three cases of how x and y may be related by profile relations.

($x \rightsquigarrow y$) It holds $x\ F^+\ y$ and $x \not F^+ y$, such that there is a path $\pi_N(x, y)$. If any path $\pi_N(x, y)$ is forwards conflict-free, this yields $x \gg y$ according to Lemma 5.1.7, a contradiction with our assumption. If there is no path $\pi_N(x, y)$ that is forwards conflict-free, there is a place $p \in P$ with $p \in \pi_N\{x, y\}$ for some $\pi_N(x, y)$, such that $|p\bullet| > 1$. If $y \in p\bullet$, we know that there is a transition $t_y \in p\bullet$ with

$t_y \neq y$ and $t_y \not F^+ y$. Thus, it holds $t_y + y$. From $x\ F^+\ t_y$, we get $x \not\# t_y$, a contradiction. If $y \notin p\bullet$, let $t_1 \in T$ be a transition with $t_1 \in p\bullet$. We know $x\ F^+\ t_1$ and, therefore, $x \not\# t_1$. As $y + t_1$ would imply $x + t_1$, we derive $y \not\# t_1$. Thus, it holds either $t_1 \rightsquigarrow y$, $t_1 \rightsquigarrow^{-1} y$, or $t_1 \| y$.

($t_1 \rightsquigarrow^{-1} y$) It holds $y\ F^+\ t_1$ and $t_1 \not F^+ y$. As $p \in \pi_N\{x, y\}$, we have $p\ F^+\ y$. Thus, there must be a transition $t_2 \in p\bullet$ with $t_2\ F^+\ y$. From $y\ F^+\ t_1$, we get $y\ F^+\ p_1$ for some $p_1 \in \bullet t_1$. Due to the free-choiceness of the net, t_1 and t_2 share all places in their pre-set, such that also $p_1\ F^+\ y$, which yields a contradiction with $y \not F^+ y$.

($t_1 \| y$) It holds either $y\ F^+\ t_1$ and $t_1\ F^+\ y$, or $y \|_{co} t_1$. The former is not in line with the assumption of $y \not F^+ y$. The latter is not possible either: let $M \in [N, M_i\rangle$ be a marking with $(N, M)[y\rangle$ and $(N, M)[t_1\rangle$. Due to $p\ F^+\ y$ either also $p \in \bullet y$ or the path implies a firing sequence $(N, M)[\sigma\rangle(N, M_2)$ due to soundness and free-choiceness, such that all places of the pre-set of y are marked at least twice. In both cases, the safeness property that holds for sound free-choice systems is violated.

Therefore, it holds $t_1 \rightsquigarrow y$ for all transitions $t_1 \in p\bullet$ for some $p \in P$ and $\pi_N(x, y)$ with $p \in \pi_N\{x, y\}$ and $|p\bullet| > 1$. Then, it also holds $t_1\ F^+\ y$ and $y \not F^+ t_1$ for all these transitions t_1. Now, either one path $\pi_N(t_1, y)$ is forwards conflict-free, which yields $t_1 \gg y$ according to Lemma 5.1.7, or there is a place $p_2 \in P$ with $p_2 \in \pi_N\{t_1, y\}$ for some $\pi_N(t_1, y)$, such that $|p_2\bullet| > 1$. In this case, the argument for p can be applied recursively for p_2, as for all transitions $t_2 \in p_2\bullet$ it holds $t_2\ F^+\ y$. Consequently, we arrive at $t_1 \gg y$ for all transitions $t_1 \in p\bullet$ for some $p \in P$ and $\pi_N(x, y)$ with $p \in \pi_N\{x, y\}$ and $|p\bullet| > 1$. Therefore, we deduce $x \gg y$, a contradiction.

($x \rightsquigarrow^{-1} y$) The argument for the case ($x \rightsquigarrow y$) is followed in reverse direction leading to a contradiction.

($x \| y$) It holds either $x\ F^+\ y$ and $y\ F^+\ x$, or $x \|_{co} y$. Again, the former is not in line with the assumption of $y \not F^+ y$. Assume that it holds $x \|_{co} y$ and consider two cases: whether there is a path $\pi_N(i, y)$ that is forwards conflict-free. If so, all firing sequences starting in M_i contain transition y or lead to a marking in which y is not dead. Hence, the assumption of $x \not\gg y$ is violated. If not, there is a place $p \in P$ with $p \in \pi_N\{i, y\}$ for some $\pi_N(i, y)$, such that $|p\bullet| > 1$. For such a place p, we prove two properties.

1. If $p\ F^+\ x$, then for all transitions $t_1 \in p\bullet$ it holds $t_1 \not F^+ y \Rightarrow t_1 \not F^+ x$. Assume that this implication does

not hold, i. e., there is a transition $t_1 \in p\bullet$ with $t_1 \not F^+ y$ and $t_1\ F^+\ x$. From $p\ F^+\ y$ we know that there must be a transition $t_2 \in p\bullet$ with either $t_2 = y$ or $t_2\ F^+\ y$. The former leads to $t_1 + y$ due to $y \not F^+ y$. Therefore, it holds $t_1 + x$, yielding a contradiction with $t_1\ F^+\ x$. If $t_2\ F^+\ y$, we know $y \not F^+ p$ from $y \not F^+ y$. Further, $y \not F^+ p$ implies $y \not F^+ t_1$. Hence, it holds either $y + t_1$ or $y \| t_1$. The latter implies the existence of a marking $M \in [N, M_i\rangle$ with $(N, M)[y\rangle$ and $(N, M)[t_1\rangle$. With $p\ F^+\ y$, this violates safeness of sound free-choice systems. Therefore, it holds $y + t_1$ and $x + t_1$, a contradiction with $t_1\ F^+\ x$.

2. If $p \not F^+ x$, then for all transitions $t_1 \in p\bullet$ it holds $t_1\ F^+\ y$. Assume that this is not the case, i. e., there is a transition $t_1 \in p\bullet$ with $t_1 \not F^+ y$. From $y \not F^+ y$ we get $y \not F^+ p$ and, therefore, $y \not F^+ t_1$. As for the previous property, $y \| t_1$ violates safeness of the system, so that it holds $y + t_1$. From $p \not F^+ x$, we get $t_1 \not F^+ x$, and $x \not F^+ t_1$ holds as well to satisfy $x \not F^+ y$. Hence, it holds either $t_1 + x$ or $t_1 \| x$. Since $x \|_{co} y$, there is a marking $M \in [N, M_i\rangle$ with $(N, M)[y\rangle$ and $(N, M)[x\rangle$. Hence, there is also a marking $M \in [N, M_i\rangle$ with $(N, M)[t_1\rangle$ and $(N, M)[x\rangle$, as $p\ F^+\ y$ and $t_1 \in p\bullet$. Therefore, it holds $t_1 \| x$, which yields a contradiction since $t_1 + y$ requires $t_1 + x$.

Now, consider all places p on a path $\pi_N(i, y)$ that are conflicts, $|p\bullet| > 1$. If $p\ F^+\ x$, the first property ensures that if y will not be part of the firing sequence due to firing of $t_1 \in p\bullet$ with $t_1 \not F^+ y$, x cannot be part either, that is, $t_1 \not F^+ x$ holds true. We also know that x and y are enabled concurrently in a marking reachable from the initial marking. Thus, once there is a conflict at place p on a path $\pi_N(i, y)$ and $p \not F^+ x$, it has to be ensured that y is not dead in a marking that marks place p. Here, the second property guarantees $t_1\ F^+\ y$ for all transitions $t_1 \in p\bullet$. That, in turn, implies $t_2 \gg t_1$ for all transitions $t_2 \in \bullet p$ and, as the property holds for all respective places p, also $t_2 \gg y$. Consequently, it holds $x \gg y$, a contradiction with our assumption.

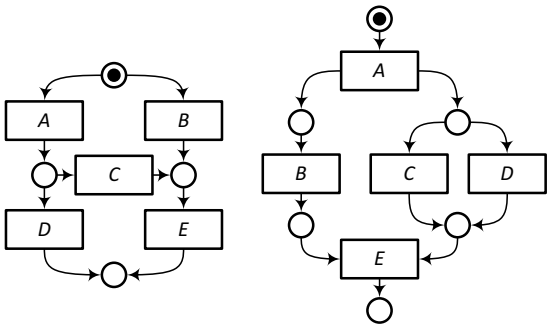
$\Rightarrow$ Let $x \gg y$ and assume that there is a transition $t \in T$ with $t + y$ and $t \not\# x$. Due to $t \not\# x$, there is a firing sequence σ with $(N, M_i)[\sigma\rangle(N, M_o)$ that contains both transitions, t and x. From $x \gg y$, we know that also $y \in \sigma$. Then, $x, y, t \in \sigma$ is a contradiction with the assumption of $t + y$.

□

Based on these results, computation of the causal behavioural profile is efficient for the respective system classes.

Corollary 5.1.11. *The following problem can be solved in* $O(n^3)$ *time with* n *as the number of transitions and places of the system:*
For a sound WF-system that is a T- or S-system, or free-choice and acyclic, to compute its causal behavioural profile.

Proof. Given any sound free-choice WF-system, the behavioural profile is computed in $O(n^3)$ time with n as the number of nodes of the system according to Corollary 5.1.6. For a sound T-system, the co-occurrence relation is set directly by Theorem 5.1.8. For a sound S-system, dominators and post-dominators are determined in linear time to the size of the net [20, 71]. Then, co-occurrence is decided according to Theorem 5.1.9. For a sound acyclic free-choice WF-system, co-occurrence is traced back to exclusiveness by Theorem 5.1.10. This requires iteration over the Cartesian product of transitions and analysis of all other transitions for each transition pair. Hence, it requires $O(n^3)$ time with n as the number of transitions. The overall time complexity is $O(n^3)$ with n as the number of nodes of the system. □



(a) (b)

Figure 27: Sound free-choice WF-systems that illustrate the derivation of the co-occurrence relation.

We illustrate the presented results with the net systems depicted in Figure 27. The system in Figure 27a is a sound workflow S-system. Therefore, the co-occurrence relation is traced back to dominators and post-dominators. For instance, transition A is a dominator of transition C and transition E is a post-dominator of transition B. Hence, we conclude $C \gg A$ and $B \gg E$ for the system in Figure 27a. The system in Figure 27b is an acyclic sound free-choice WF-system. As an example, consider transitions A and C. Both are not exclusive. There exists a transition, namely D, that is exclusive to C but not to A. Hence, the pair (A, C) is not in the co-occurrence relation of the system in Figure 27b. The opposite, $C \gg A$, holds true, as there is no transition that is exclusive to transition A.

5.2 COMPUTATIONS USING STRUCTURAL DECOMPOSITION

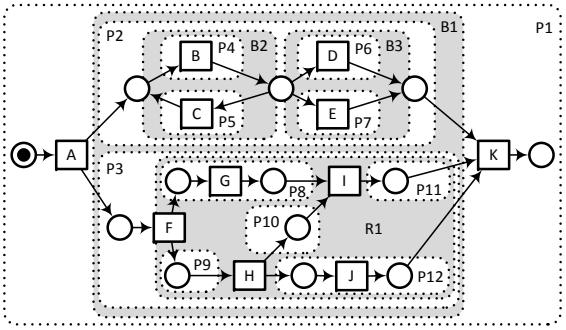
The results introduced in the previous section allow for computing the behavioural profile for the Cartesian product of transitions. Depending on the concrete use case, however, various transitions may be irrelevant for analysis. If so, computation of the profile relations for all pairs of transitions leads to a computational overhead. Therefore, this section introduces an alternative approach for the computation of behavioural profiles for the class of sound free-choice WF-systems. By leveraging structural decomposition techniques, the relations of the causal behavioural profile are determined for a single pair of transitions in linear time to the size of the net. The technique assumes both transitions to be part of a structured region of the WF-system, i. e., a region which shows a hierarchy of single-entry-single-exit (SESE) subnets. Extending the results presented before, it allows for the computation of co-occurrence for transitions in cyclic sound free-choice WF-system, if the circuits are well-structured.

In the remainder of this section, first, we show how an existing decomposition technique, the Refined Process Structure Tree (RPST) [481, 359], is applied to WF-nets. Second, we introduce behavioural annotations for the RPST of sound free-choice WF-systems. This yields the notion of a WF-tree. Third, we establish the relation between the WF-tree of a net system and its causal behavioural profile. The section closes with the discussion of a computation algorithm for the derivation of causal behavioural profiles that combines the results obtained by structural decomposition with those introduced in the previous section.

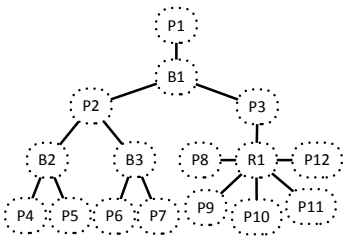
The RPST of WF-nets

The Refined Process Structure Tree (RPST) [481, 359] is a technique for detecting the structure of a workflow graph. Such a graph represents the control flow structure of a process model by means of activities and routing elements. A workflow graph can be parsed into a hierarchy of *fragments* with a single entry and a single exit, such that the RPST is a containment hierarchy of *canonical* fragments of the graph. The RPST is unique for a given workflow graph and can be computed in linear time [481, 359]. Although the RPST has been introduced for workflow graphs, it can be applied to other graph based models such as WF-nets in a straight-forward manner. We define basic terms of the RPST for WF-nets as follows.

Definition 5.2.1 (Flows, Entry, Exit, Canonical Fragment)
Let $N = (P, T, F)$ be a WF-net and $X = (P \cup T)$ its set of nodes.



(a) WF-system and its canonical fragments.



(b) The RPST of (a).

Figure 28: A WF-system, its canonical fragments, and its RPST.

- For a node $x \in X$ of a net $N = (P, T, F)$, $in_N(x) = \{(n, x) \in F \mid n \in {}^\bullet x\}$ are its *incoming flows* and $out_N(x) = \{(x, n) \in F \mid n \in x^\bullet\}$ are its *outgoing flows*.
- A node $x \in X'$ of a connected subnet $N' = (P', T', F')$ of a net N is a *boundary node*, iff $\exists\ e \in in_N(x) \cup out_N(x)\ [\ e \notin F'\ ]$. If x is a boundary node, it is an *entry* of N', iff $in_N(x) \cap F' = \emptyset$ or $out_N(x) \subseteq F'$, or an *exit* of N', iff $out_N(x) \cap F' = \emptyset$ or $in_N(x) \subseteq F'$.
- Any connected subnet ω of N is a *fragment*, iff it has exactly two boundary nodes, one entry and one exit denoted by $\omega_\triangleleft$ and $\omega_\triangleright$, respectively.
- A fragment is *place bordered* (*transition bordered*), iff its boundary nodes are places (transitions).
- A fragment $\omega = (P_\omega, T_\omega, F_\omega)$ is *canonical* in a set of *all* fragments Σ of N, iff $\forall\ \gamma = (P_\gamma, T_\gamma, F_\gamma) \in \Sigma\ [\ \omega \neq \gamma \Rightarrow (F_\omega \cap F_\gamma = \emptyset) \vee (F_\omega \subset F_\gamma) \vee (F_\gamma \subset F_\omega)\ ]$.

Figure 28 exemplifies the RPST for a WF-system. Figure 28a illustrates the canonical fragments of a WF-system, each of them formed by a set of flows enclosed in or intersecting the region with a dotted border. Figure 28b provides an alternative view, where each node represents a canonical fragment and edges hint at the containment of fragments. A tree structure is obtained in this view – the RPST. For instance, fragment B1 has two boundary transitions: entry A and exit K, is contained in fragment P1, and contains fragments P2 and P3. Note that trivial fragments are not visualised.

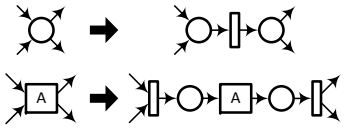


Figure 29: Node-splitting for normalisation.

If the RPST is computed for a *normalised* workflow graph, i. e., a workflow graph that does not contain nodes with multiple incoming and multiple outgoing flows, each canonical fragment can be classified to one out of four structural classes [357, 359]: A *trivial* (T) fragment consists of a single flow. A *polygon* (P) represents a sequence of nodes (fragments). A *bond* (B) stands for a collection of fragments that share common boundary nodes. Any other fragment is a *rigid* (R). We use fragment names that hint at their structural class, e. g., R1 is a rigid fragment. Every workflow graph can be normalised by performing a node-splitting pre-processing, illustrated for WF-nets in Figure 29. The WF-system in Figure 28a is normalised.

The Notion of a WF-tree

The fragments derived by the RPST can be related to behavioural properties of the underlying WF-system. We concretise RPST fragments by annotating them with behavioural characteristics. The containment hierarchy of annotated canonical fragments of a WF-system is referred to as the *WF-tree*. The WF-tree is defined for sound free-choice WF-systems as this class of systems shows a tight coupling of syntax and semantics, cf., Section 5.1.

Definition 5.2.2 (WF-Tree)
Let (N, M_i) be a sound free-choice WF-system. The *WF-Tree* of N is a tuple $\mathcal{T}_N = (\Omega, \chi, t, b)$, where:

- Ω is a set of *all* canonical fragments of N,
- $\chi : \Omega \to \wp(\Omega)$ is a function that assigns child fragments to fragments, such that it holds $\forall\ \omega, \gamma \in \Omega\ [\ (\chi(\omega) \cap \chi(\gamma) \neq \emptyset) \Rightarrow \omega = \gamma\ ]$.
- $t : \Omega \to \{T, P, B, R\}$ is a function that assigns a type to a fragment,
- $b : \Omega_B \to \{B_\circ, B_\diamond, L\}$, $\Omega_B = \{\omega \in \Omega \mid t(\omega) = B\}$, is a function that assigns a refined type to a bond fragment, where $B_\circ$, $B_\diamond$, and L types stand for place bordered, transition bordered, and loop bonds, respectively.

Further, we define auxiliary concepts for the WF-tree.

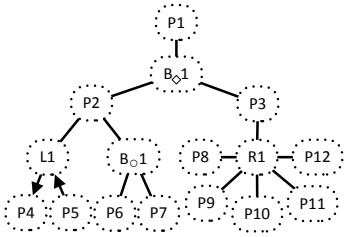


Figure 30: The WF-tree for the system in Figure 28.

Definition 5.2.3 (Parent, Child, Root, Ancestor, LCA, Path)
Let $\mathcal{T}_N = (\Omega, \chi, t, b)$ be a WF-tree.

- For any fragment $\omega \in \Omega$, ω is a *parent* of γ and γ is a *child* of ω, iff $\gamma \in \chi(\omega)$. χ^+ is the irreflexive transitive closure of χ.
- The fragment $\omega \in \Omega$ is a *root* of $\mathcal{T}$, denoted by ω_r, iff it has no parent.
- The partial function $\rho : \Omega \setminus \{\omega_r\} \to \Omega$ assigns parents to fragments.
- For any fragment $\omega \in \Omega$, ω is an *ancestor* of ϑ and ϑ is a *descendant* of ω, if $\vartheta \in \chi^+(\omega)$.
- For any two fragments $\omega, \gamma \in \Omega$ their *Lowest Common Ancestor* (LCA), denoted by $\mathtt{lca}(\omega, \gamma)$, is the shared ancestor of ω and γ that is located farthest from the root of the WF-tree. By definition, $\mathtt{lca}(\omega, \omega) = \omega$.
- For any fragment $\omega_1 \in \Omega$ and its descendant $\omega_n \in \Omega$, a *downward path* from ω_1 to ω_n, denoted by $\pi_{\mathcal{T}}(\omega_1, \omega_n)$, is a sequence $\pi_{\mathcal{T}} = \langle \omega_1, \ldots, \omega_n \rangle$, such that ω_i is a parent of ω_{i+1} for all $i \in \mathbb{N}$, $1 \leqslant i < n$. Further, we define $\pi_{\mathcal{T}}(\omega_1, \omega_n, i) = \omega_i$ to refer to an entry of the path and $\pi_{\mathcal{T}}\{\omega_1, \omega_n\} = \{\omega_1, \ldots, \omega_n\}$ to refer to all nodes on the path.

Figure 30 shows the WF-tree of the WF-system from Figure 28. The WF-tree is isomorphic to the RPST of the WF-system. Given the RPST, adding the behavioural annotation is a trivial task for most fragments, except of the following cases: A bond fragment $\gamma = (P_\gamma, T_\gamma, F_\gamma) \in \mathrm{dom}(b)$ of $\mathcal{T}_N = (\Omega, \chi, t, b)$ is assigned the L type, if $\gamma_\triangleleft = \omega_\triangleright$ with ω being a child of γ. Otherwise, $b(\gamma) = B_\circ$ if $\gamma_\triangleleft \in P_\gamma$, or $b(\gamma) = B_\diamond$ if $\gamma_\triangleleft \in T_\gamma$.

Children of a polygon fragment are arranged with respect to their execution order. A partial function $\mathtt{order} : \Omega' \to \mathbb{N}_0$, $\Omega' = \{\omega \in \Omega \setminus \{\omega_r\} \mid t(\rho(\omega)) = P\}$ assigns to children of polygon fragments their respective order positions; $\mathtt{order}(\omega) = 0$, if $\omega_\triangleleft = \gamma_\triangleleft$ with $\gamma = \rho(\omega)$ being the parent, and $\mathtt{order}(\omega) = i$, $i \in \mathbb{N}$, if $\omega_\triangleleft = \vartheta_\triangleright$ for some $\vartheta \in \Omega$, such that $\mathtt{order}(\vartheta) = i - 1$. The orders of two nodes are only comparable if they share a common parent. For instance, in Figure 30, $\mathtt{order}(L1) = 1$ and $\mathtt{order}(B_\circ 1) = 2$. This means that the transitions of fragment L1 are located before

the transitions of fragment $B_\circ 1$ inside polygon P2. The layout of child fragments of polygon fragments in Figure 30 hints at their order relation.

Children of a loop fragment are classified as *forward* ($\Rightarrow$) or *backward* ($\Leftarrow$). A partial function $\ell : \Omega'' \to \{\Leftarrow, \Rightarrow\}$ with $\Omega'' = \{\omega \in \Omega \setminus \{\omega_r\} \mid b(\rho(\omega)) = L\}$ assigns an orientation to children of loop fragments. $\ell(\omega) = \Rightarrow$ if $\omega_\triangleleft = \gamma_\triangleleft$ with $\gamma = \rho(\omega)$, otherwise $\ell(\omega) = \Leftarrow$. In Figure 30, P4 and P5 are forward and backward fragments, which is visualised by the direction of edges.

We introduce two lemmas that prove the completeness of the codomain of function b by showing that a bond fragment is either place or transition bordered, and that each loop fragment is place bordered. A rigid fragment bordered with a place and a transition can still be free-choice and sound. An example for such a net structure can be found in [455].

The following results leverage the notions of a handle and a bridge. We shortly recall these notions following on [150]. For a net $N = (P, T, F)$ and a partial subnet N' a path $\pi_N(x_1, x_k)$ of N, $k > 1$ and all nodes x_i on the path are distinct, is a *handle* of N', if and only if $\pi_N \cap (P' \cup T') = \{x_1, x_k\}$. For a net $N = (P, T, F)$ and two partial subnets N', N'' a path $\pi_N(x_1, x_k)$ of N, $k > 1$ and all nodes x_i on the path are distinct, is a *bridge* from N' to N'', if and only if $\pi_N \cap (P' \cup T') = \{x_1\}$ and $\pi_N \cap (P'' \cup T'') = \{x_k\}$. We speak of PP-, TT-, PT-, TP-handles and bridges, depending on the type, place or transition, of the initial and the final node of the respective path.

Lemma 5.2.1. *Let $\mathcal{T}_N = (\Omega, \chi, t, b)$ be the WF-tree of a sound free-choice WF-system (N, M_i), $N = (P, T, F)$. No bond fragment $\omega \in \Omega$, $t(\omega) = B$, has $\{p, t\}$ boundary nodes, where $p \in P$ and $t \in T$.*

Proof. Assume ω is a bond fragment with $\{p, t\}$ boundary nodes. There exists a circuit Γ in a short-circuit net of N that contains $\{p, t\}$. Let Γ_ω be a subpath of Γ inside ω. There exists a child fragment γ of ω that contains Γ_ω. A bond fragment has $k \geqslant 2$ child fragments [357, 359]. Let ϑ be a child of ω, $\vartheta \neq \gamma$. We distinguish two cases:

- Let H be a path from p to t contained in ϑ. H is a PT-handle of Γ. In a live and bounded free-choice system, H is bridged to Γ_ω through a TP-bridge K, see Proposition 4.2 in [150]. This implies that $\vartheta = \gamma$; otherwise bond fragment ω contains path K that is not inside of a single child fragment [357, 359]. Thus, ω has a single child fragment, a contradiction with the assumption of ω being a bond fragment.
- Let H be a path from t to p contained in ϑ. H is a TP-handle of Γ. In a live and bounded free-choice system, no circuit has TP-handles, see Proposition 4.1 in [150], which yields a contradiction with our assumptions.

□

Lemma 5.2.2. *Let $\mathcal{T}_N = (\Omega, \chi, t, b)$ be the WF-tree of a sound free-choice WF-system, (N, M_i), $N = (P, T, F)$. A loop fragment $\omega = (P_\omega, T_\omega, F_\omega) \in \Omega$, $b(\omega) = L$, is place bordered, i. e., $\{\omega_\triangleleft, \omega_\triangleright\} \subseteq P$.*

Proof. Because of Lemma 5.2.1, ω is either place or transition bordered. Assume ω is transition bordered. There exists place $p \in P$ such that $p \in (\bullet\omega_\triangleleft \cap P_\omega)$, $M_i(p) = 0$. Transition $\omega_\triangleleft$ is enabled if there exists a marking $M \in [N, M_i\rangle$ with $M(p) > 0$. Since ω is a connected subnet, for all transitions $t \in T_\omega \setminus \{\omega_\triangleleft, \omega_\triangleright\}$ all flows are in ω, i. e., $(in_N(t) \cup out_N(t)) \subseteq F_\omega$. Thus, every path from i to p visits $\omega_\triangleleft$. $M(p) > 0$ is only possible, if $\omega_\triangleleft$ has fired before. We reached a contradiction. Transition $\omega_\triangleleft$ is never enabled and N is not live, and hence, not sound. Since any loop fragment is not transition bordered, it is place bordered, cf., Lemma 5.2.1. □

For sound free-choice WF-systems, the WF-tree can be derived efficiently.

Proposition 5.2.3. *The following problem can be solved in linear time. Given a sound free-choice WF-system, to compute its WF-tree.*

Proof. Given a workflow graph, its RPST can be computed in time linear to the number of flows of the graph [359, 481]. The number of canonical fragments in the RPST is linear to the number of flows in the workflow graph [37, 190, 359]. Given the RPST of a WF-system, we iterate over all bond fragments and assign the behavioural annotations. Here, it suffices to check the type of the entry node, either a place or transition, and to determine whether the entry is also the exit of a child fragment. That can be decided in constant time for each fragment. Finally, child fragments of a polygon can be ordered in linear time. We introduce a hash function that returns a child fragment with the given node as an entry and iterate over the children of the polygon. □

Computations based on the WF-tree

For the computation of the causal behavioural profile for a pair of transitions based on the WF-tree, we assume that each transition has one incoming and one outgoing flow arc. If this is not the case, we apply the pre-processing illustrated in Figure 31, which preserves the behaviour of the system [329] and, therefore, does not change the causal behavioural profile. Let (N, M_i) be a WF-system with $N = (P, T, F)$ and $\mathcal{T}_N = (\Omega, \chi, t, b)$ the WF-tree of N. After pre-processing, each transition $t \in T$ is a boundary node of at most two trivial fragments of $\mathcal{T}_N$. Consequently,

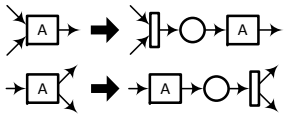


Figure 31: Pre-processing for structural decomposition.

it suffices to show how the relations of the causal behavioural profile are determined for the entries of two trivial fragments.

In the absence of rigid fragments, the profile relations are computed as follows. For the self-relation of two transitions, we check whether there is a bond fragment of type loop that comprises both trivial fragments. For two distinct transitions, strict order is derived if the LCA of the respective trivial fragments is a polygon fragment and the transitions are not part of a circuit. Exclusiveness holds between two transitions, if the LCA of the respective trivial fragments is place-bordered acyclic bond fragment and the transitions are not part of a circuit. If the transitions are part of a circuit or if the according LCA is a transition-bordered bond fragment, we conclude on interleaving order.

Theorem 5.2.4. *Let* $\mathcal{T}_N = (\Omega, \chi, t, b)$ *be a WF-tree,* $\alpha, \beta \in \Omega$ *two trivial fragments,* $\gamma = \mathrm{lca}(\alpha, \beta)$*, and* $\forall\ \omega \in \pi_{\mathcal{T}}\{\omega_r, \gamma\}\ [\ t(\omega) \neq R\ ]$.

1. *If* $\alpha = \beta$*, then* $\alpha_\triangleleft \| \beta_\triangleleft$*, iff* $\exists\ \omega \in \pi_{\mathcal{T}}\{\omega_r, \gamma\}\ [\ b(\omega) = L\ ]$*. Otherwise,* $\alpha_\triangleleft + \beta_\triangleleft$.
2. *If* $\alpha \neq \beta$*, then it holds:*
   - $\alpha_\triangleleft \rightsquigarrow \beta_\triangleleft$*, if and only if (1)* $t(\gamma) = P \wedge \mathrm{order}(\pi_{\mathcal{T}}(\gamma, \alpha, 1)) < \mathrm{order}(\pi_{\mathcal{T}}(\gamma, \beta, 1))$*, and (2)* $\forall\ \omega \in \pi_{\mathcal{T}}\{\omega_r, \gamma\}\ [b(\omega) \neq L]$.
   - $\alpha_\triangleleft + \beta_\triangleleft$*, if and only if (1)* $b(\gamma) = B_\circ$*, and (2)* $\forall\ \omega \in \pi_{\mathcal{T}}\{\omega_r, \gamma\}\ [\ b(\omega) \neq L\ ]$.
   - $\alpha_\triangleleft \| \beta_\triangleleft$*, if and only if (1)* $b(\gamma) \in \{B_\diamond, L\}$*, or (2)* $\exists\ \omega \in \pi_{\mathcal{T}}\{\omega_r, \gamma\}\ [\ b(\omega) = L\ ]$.

Proof. Let $\mathcal{T}_N$, α, β, γ be defined as before, (N, M_i) the respective WF-system, and $\forall\ \omega \in \pi_{\mathcal{T}}\{\omega_r, \gamma\}\ [\ t(\omega) \neq R\ ]$.

1. Let $\alpha = \beta$.

$\Rightarrow$ Let $\alpha_\triangleleft \| \beta_\triangleleft$ and assume $\forall\ \omega \in \pi_{\mathcal{T}}\{\omega_r, \gamma\}\ [\ b(\omega) \neq L\ ]$. Due to $\alpha = \beta$, also $\alpha_\triangleleft = \beta_\triangleleft$. Thus, it holds $\alpha_\triangleleft \| \alpha_\triangleleft$. Due to safeness of (N, M_i), $\alpha_\triangleleft \| \alpha_\triangleleft$ cannot be traced back to concurrent enabling of $\alpha_\triangleleft$. According to Lemma 5.1.2, this implies $\alpha_\triangleleft\ F^+\ \alpha_\triangleleft$. Circuits are part of B (if the bond is a loop fragment) or R type fragments. Thus, there has to be a fragment ω, which is an ancestor of α and $t(\omega) = R$ or $b(\omega) = L$. As the LCA of α is $\gamma = \alpha$ by definition, this yields a contradiction with the assumptions.

$\Leftarrow$ Let $\exists\ \omega \in \pi_{\mathcal{T}}\{\omega_r, \gamma\}\ [\ b(\omega) = L\ ]$ and assume $\alpha_\triangleleft \nparallel \beta_\triangleleft$. Since $\alpha = \beta$, it holds $\alpha_\triangleleft = \beta_\triangleleft$. One of the ancestors of α is a B type

fragment that is a loop, such that $\alpha_\lhd \; F^+ \; \alpha_\lhd$. Since (N, M_i) is safe, $\alpha_\lhd$ cannot be enabled concurrently with itself, so that $\alpha_\lhd \| \alpha_\lhd$ by Lemma 5.1.2.

2. Let $\alpha \neq \beta$.

$\Rightarrow$ Let $\alpha_\lhd \rightsquigarrow \beta_\lhd$ and assume (1) $\mathrm{order}(\pi_\mathcal{T}(\gamma, \alpha, 1)) > \mathrm{order}(\pi_\mathcal{T}(\gamma, \beta, 1))$ or $t(\gamma) \neq P$, or (2) $\exists\, \omega \in \pi_\mathcal{T}\{\omega_r, \gamma\}\, [\, b(\omega) = L \,]$. According to Lemma 5.1.4, $\alpha_\lhd \rightsquigarrow \beta_\lhd$ implies $\alpha_\lhd \; F^+ \; \beta_\lhd$ and $\beta_\lhd \; \not{F^+} \; \alpha_\lhd$. Thus, assumption (2) cannot hold as an L type fragment that is an ancestor of both, α and β, would imply $\beta_\lhd \; F^+ \; \alpha_\lhd$. The first part of assumption (1) cannot hold either: $b(\gamma) = L$ contradicts with the flow dependencies between $\alpha_\lhd$ and $\beta_\lhd$, whereas $t(\gamma) = R$, $t(\gamma) = B$ and $b(\gamma) \in \{B_\circ, B_\diamond\}$, and $t(\gamma) = T$ (which would imply $\alpha = \beta$) disqualify due to our assumptions. Thus, it holds $t(\gamma) = P$. The order in a P type fragment coincidences with the flow dependencies, i. e., $\alpha_\lhd \; F^+ \; \beta_\lhd$, which yields a contradiction.

Let $\alpha_\lhd + \beta_\lhd$ and assume (1) $b(\gamma) \neq B_\circ$ or (2) $\exists\, \omega \in \pi_\mathcal{T}\{\omega_r, \gamma\}\, [\, b(\omega) = L \,]$. According to Lemma 5.1.3, the former implies $\alpha_\lhd \; \not{F^+} \; \beta_\lhd$ and $\beta_\lhd \; \not{F^+} \; \alpha_\lhd$. That, in turn, implies that assumption (2) cannot hold and $\gamma \neq P$. Since $\gamma \neq R$ and $\gamma \neq T$ (which would imply $\alpha = \beta$), we conclude $t(\gamma) = B$. As the flow dependencies preclude $b(\gamma) = L$, we assume $b(\gamma) = B_\diamond$. Then, $\gamma_\lhd$ is a transition. Due to soundness, there are two markings $M_1, M_2 \in [N, M_i\rangle$, such that $(N, M_1)[\gamma_\lhd\rangle(N, M_2)$. As γ is an ancestor of both, α and β, we know $\gamma_\lhd \; F^+ \; \alpha_\lhd$ and $\gamma_\lhd \; F^+ \; \beta_\lhd$. That implies that both transitions, $\alpha_\lhd$ and $\beta_\lhd$, might get enabled in a firing sequences starting in M_2. That is not in line with $\alpha_\lhd + \beta_\lhd$. Thus, $b(\gamma) = B_\circ$, a contradiction with assumption (1).

Let $\alpha_\lhd \| \beta_\lhd$ and assume (1) $b(\gamma) = B_\circ$ and (2) $\forall\, \omega \in \pi_\mathcal{T}\{\omega_r, \gamma\}\, [\, b(\omega) \neq L \,]$. According to Lemma 5.1.2, $\alpha_\lhd \| \beta_\lhd$ implies concurrent enabling of a both transitions in a certain marking, or $\alpha_\lhd \; F^+ \; \beta_\lhd$ and $\beta_\lhd \; F^+ \; \alpha_\lhd$. The latter is not possible due to assumption (2). Thus, we assume concurrent enabling. Let $x \in \gamma_\lhd\bullet$ be a successor of $\gamma_\lhd$. γ is the LCA of α and β. Consequently, $x \; F^+ \; \alpha_\lhd$ implies $x \; \not{F^+} \; \beta_\lhd$ and vice versa. Thus, concurrent enabling of $\alpha_\lhd$ and $\beta_\lhd$ requires $\gamma_\lhd$ to be a transition. That is a contradiction with assumption (1).

$\Leftarrow$ Let (1) $t(\gamma) = P \;\wedge\; \mathrm{order}(\pi_\mathcal{T}(\gamma, \alpha, 1)) < \mathrm{order}(\pi_\mathcal{T}(\gamma, \beta, 1))$, and (2) $\forall\, \omega \in \pi_\mathcal{T}\{\omega_r, \gamma\}\, [\, b(\omega) \neq L \,]$ and assume $\alpha_\lhd \not\rightsquigarrow \beta_\lhd$. From (1) and (2), we conclude $\alpha_\lhd \; F^+ \; \beta_\lhd$ and $\beta_\lhd \; \not{F^+} \; \alpha_\lhd$, which is equivalent to $\alpha_\lhd \rightsquigarrow \beta_\lhd$ by Lemma 5.1.4.

Let (1) $b(\gamma) = B_\circ$ and (2) $\forall\, \omega \in \pi_\mathcal{T}\{\omega_r, \gamma\}\, [\, b(\omega) \neq L \,]$, and assume $\alpha_\lhd \not\| \beta_\lhd$. From (1) and (2), we get $\alpha_\lhd \; \not{F^+} \; \beta_\lhd$ and $\beta_\lhd \; \not{F^+} \; \alpha_\lhd$. Therefore, we assume that both transitions are enabled concurrently. Due to $b(\gamma) = B_\circ$, $\gamma_\lhd$ is a place. Let $t \in \gamma_\lhd\bullet$ be a successor of $\gamma_\lhd$. Due to soundness, there are two

markings $M_1, M_2 \in [N, M_i\rangle$, such that $(N, M_1)[t\rangle(N, M_2)$. As γ is the LCA of both, we know that $t\ F^+\ \alpha_\triangleleft$ implies $t\ \not{F^+}\ \beta_\triangleleft$, and vice versa. Thus, any firing sequence starting in M_2 contains either $\alpha_\triangleleft$, $\beta_\triangleleft$, or none of the two transitions. As γ is the parent of both, α and β, $\gamma_\triangleleft$ is on every path from the initial place i to $\alpha_\triangleleft$ and $\beta_\triangleleft$. Therefore, there does not exist a firing sequences containing both transitions, which leads to $\alpha_\triangleleft + \beta_\triangleleft$.
Let (1) $b(\gamma) \in \{B_\diamond, L\}$ or (2) $\exists\ \omega \in \pi_\mathcal{T}\{\omega_r, \gamma\}\ [\ b(\omega) = L\ ]$, and assume $\alpha_\triangleleft \nparallel \beta_\triangleleft$. From requirement (2), we get $\alpha_\triangleleft\ F^+\ \beta_\triangleleft$ and $\beta_\triangleleft\ F^+\ \alpha_\triangleleft$. According to Lemma 5.1.2, this is equivalent to $\alpha_\triangleleft \| \beta_\triangleleft$, which is not in line with our assumption. The same holds true for $b(\gamma) = L$. Consider $b(\gamma) = B_\diamond$. Then, $\gamma_\triangleleft$ is a transition. Let $p_1, p_2 \in \gamma_\triangleleft\bullet$ be two successors of $\gamma_\triangleleft$ with $p_1\ F^+\ \alpha_\triangleleft$ and $p_2\ F^+\ \beta_\triangleleft$. The existence of these paths implies the existence of a firing sequence, i. e., $\alpha_\triangleleft$ and $\beta_\triangleleft$ can get enabled concurrently. That, in turn, is equivalent to $\alpha_\triangleleft \| \beta_\triangleleft$ by Lemma 5.1.1, yielding a contradiction.

□

The co-occurrence relation of the causal behavioural profile is computed in the absence of rigid fragments as follows. Given a pair of transitions, we obtain the LCA of the respective trivial fragments. Then, we investigate fragments on the path from the LCA to the trivial fragment that is related to the second transition of the co-occurrence dependency. Co-occurrence holds, if these fragments are of type polygon, transition-bordered bond, or loop bond. For loop fragments, we also check whether they show a single path from the entry to the exit that contains the second transition of the co-occurrence dependency. Only in this case, co-occurrence holds from the first to the second transition.

Theorem 5.2.5. *Let $\mathcal{T}_N = (\Omega, \chi, t, b)$ be a WF-tree and $\alpha, \beta \in \Omega$ two trivial fragments, $\alpha \neq \beta$. Let $\gamma = lca(\alpha, \beta)$, $\Pi = \pi_\mathcal{T}\{\gamma, \beta\}$, and $\forall\ \omega \in \Pi\ [\ t(\omega) \neq R\ ]$. Then, $\alpha_\triangleleft \gg \beta_\triangleleft$, iff for all fragments $\omega \in (\Pi \setminus \{\beta\})$ one of the following conditions holds:*

1. $t(\omega) = P$,
2. $t(\omega) = B$ *and* $b(\omega) = B_\diamond$, *or*
3. $t(\omega) = B$, $b(\omega) = L$, *and with* $\Theta = \{\vartheta \in \chi(\omega) \mid \ell(\vartheta) \Longrightarrow\}$ *it holds* $\forall\ \vartheta \in \Theta\ [\ \beta \in \chi^+(\vartheta)\ ]$.

Proof. Let $\mathcal{T}_N$, α, β, γ, Π be defined as before, (N, M_i) the respective WF-system, and $\forall\ \omega \in \Pi\ [\ t(\omega) \neq R\ ]$. For both directions of the proof, let $\delta = \rho(\beta)$ and $\eta = \rho(\delta)$ be the parents of β and δ. Note that we know $t(\delta) = P$ and $t(\eta) \notin \{R, T\}$.

⇒ Let $\alpha_\triangleleft \gg \beta_\triangleleft$ and assume that there is a fragment $\omega \in (\Pi \setminus \{\beta\})$ with $t(\omega) \neq P$ or $b(\omega) \neq B_\diamond$ or if $b(\omega) = L$ then it holds $\exists\ \vartheta \in \Theta\ [\ \beta \notin \chi^+(\vartheta)\ ]$ with $\Theta = \{\vartheta \in \chi(\omega) \mid \ell(\vartheta) \Longrightarrow\}$. For all fragments $\omega \in (\Pi \setminus \{\beta\})$, we know $t(\omega) \neq R$ and $t(\omega) \neq T$

(as $\beta \in \chi^+(\omega)$). We first consider the LCA, i. e., fragment γ. Let $\epsilon \in \chi(\gamma)$ with $\alpha \in \chi^+(\epsilon)$ be the child fragment of γ that contains α (it holds $\epsilon \neq \delta$). We distinguish two cases.

(1) $\gamma_\lhd$ is a transition. Then, $t(\gamma) \in \{P, B\}$, and $t(\gamma) = B$ requires $b(\gamma) = B_\diamond$.

(2) $\gamma_\lhd$ is a place. Then, $t(\gamma) \in \{P, B\}$, and $t(\gamma) = B$ requires $b(\gamma) \in \{B_\circ, L\}$. We distinguish two cases (I) $b(\gamma) = B_\circ$ and (II) $b(\gamma) = L$.

(I) Let $M_1, M_2 \in [N, M_i\rangle$ be markings with $M_1(\gamma_\lhd) > 1$ and $M_2(\gamma_\rhd) > 1$. Let σ_1, σ_2 be two firing sequences with $(N, M_1)[\sigma_1\rangle(N, M_2)[\sigma_2\rangle(N, M_o)$, such that σ_2 does not contain any transition that is part of γ. As fragment ϵ represents a path from $\gamma_\lhd$ to $\gamma_\rhd$, σ_1 might contain only transitions that are part of ϵ. Then, it holds $\alpha_\lhd \in \sigma_1$. Since $\alpha_\lhd \gg \beta_\lhd$, also $\beta_\lhd \in \sigma_1$ (as $\beta_\lhd \notin \sigma_2$). Therefore, $\beta \in \chi^+(\epsilon)$, such that we arrived at a contradiction with the definition of $\gamma = lca(\alpha, \beta)$.

(II) Let $M_1, M_2 \in [N, M_i\rangle$ be two markings as defined for the previous case. Consider the case of ϵ having forward orientation, $\ell(\epsilon) = \Longrightarrow$. Then, there are firing sequences σ_1, σ_2 with $(N, M_1)[\sigma_1\rangle(N, M_2)[\sigma_2\rangle(N, M_o)$, such that σ_1 contains only transitions that are part of ϵ, whereas σ_2 does not contain any transition that is part of γ. Then, $\alpha_\lhd$ might be part of σ_1. As $\beta_\lhd \notin \sigma_2$, but $\alpha_\lhd \gg \beta_\lhd$, we conclude $\beta_\lhd \in \sigma_1$. Thus, it holds $\beta \in \chi^+(\epsilon)$, which, again, yields a contradiction with the definition of $\gamma = lca(\alpha, \beta)$. Consider the case of ϵ having backward orientation, $\ell(\epsilon) = \Longleftarrow$. Then, there is a firing sequence σ_3 with $(N, M_1)[\sigma_1\rangle(N, M_2)[\sigma_3\rangle(N, M_1)$ and $(N, M_1)[\sigma_1\rangle(N, M_2)[\sigma_2\rangle(N, M_o)$, such that σ_3 contains solely transitions that are part of ϵ. Again, $\alpha_\lhd$ can be part of σ_3. From $\beta_\lhd \notin \sigma_2$ but $\alpha_\lhd \gg \beta_\lhd$, it follows $\beta_\lhd \in \sigma_1$ or $\beta_\lhd \in \sigma_3$. The latter would imply $\beta \in \chi^+(\epsilon)$ (a contradiction as earlier), which leads to $\beta_\lhd \in \sigma_1$. To ensure $\alpha_\lhd \gg \beta_\lhd$, *every* firing sequence σ_1 has to contain $\beta_\lhd$. Therefore, all children of fragment γ that represent paths from $\gamma_\lhd$ to $\gamma_\rhd$, i. e., children with forward orientation, have to contain β. As β can only be contained in one child of fragment γ, there is only one child with forward orientation.

We summarise that $b(\gamma) \neq B_\circ$, and $b(\gamma) = L$ implies that $\forall\, \vartheta \in \Theta\, [\, \beta \in \chi^+(\vartheta)\,]$ with $\Theta = \{\vartheta \in \chi(\gamma) \mid \ell(\vartheta) = \Longrightarrow\}$.

For both cases, $\gamma_\lhd$ being a transition or a place, we see that fragment γ does not satisfy the assumptions on a fragment $\omega \in (\Pi \setminus \{\beta\})$ as stated before. We now consider two cases, $\eta = \gamma$ or γ is an ancestor of η. Due to $t(\delta) = P$, the former yields a contradiction, as $\Pi \setminus \{\beta\} = \{\gamma, \delta\}$ and both fragments

do not satisfy our assumption. For γ being an ancestor of η, there is a fragment κ, such that $\kappa \in \chi(\gamma)$ and $\eta \in \chi^+(\kappa)$. Again, we distinguish two cases.

(1) $\kappa_\lhd$ is a transition. Then, $t(\kappa) \in \{P, B\}$, and $t(\kappa) = B$ requires $b(\kappa) = B_\diamond$.

(2) $\kappa_\lhd$ is a place. Now, we distinguish the three possible types of fragments for γ.

(I) If $t(\gamma) = P$, without loss of generality, we assume $\gamma_\lhd$ and $\gamma_\lhd$ to be places (the single places of their post-set or pre-set, respectively, would be taken if $\gamma_\lhd$ or $\gamma_\lhd$ would be a transition). Let $M_1, M_2 \in [N, M_i\rangle$ be markings with $M_1(\gamma_\lhd) > 1$ and $M_2(\gamma_\rhd) > 1$. Let σ_1, σ_2 be firing sequences with $(N, M_1)[\sigma_1\rangle(N, M_2)[\sigma_2\rangle(N, M_o)$, such that σ_2 does not contain any transition that is part of γ. Due to $t(\gamma) = P$, either $\kappa_\rhd \; F^+ \; \epsilon_\lhd$ and $\epsilon_\rhd \not F^+ \kappa_\lhd$, or vice versa. In both cases, $\alpha_\lhd \gg \beta_\lhd$ requires that a firing sequence σ_3 between two markings $M_3, M_4 \in [N, M_i\rangle$ with $M_3(\kappa_\lhd) > 1$ and $M_4(\kappa_\rhd) > 1$ contains $\beta_\lhd$. That is due to firing sequences leading from M_1 to M_3, or from M_4 to M_2 that contain no transition of fragment κ, but transition $\alpha_\lhd$.

(II) If $b(\gamma) = L$, we know that $\forall\, \vartheta \in \Theta\, [\, \beta \in \chi^+(\vartheta)\,]$ with $\Theta = \{\vartheta \in \chi(\gamma) \mid \ell(\vartheta) = \Rightarrow\}$. As β can only be contained in one child of fragment γ, i.e., fragment κ, we know that $\ell(\kappa) = \Rightarrow$ and, in turn, $\ell(\epsilon) = \Leftarrow$. Let M_1, M_2, σ_1, and σ_2 be defined as before. We may observe firing sequences σ_4, σ_5 with $(N, M_1)[\sigma_1\rangle(N, M_2)[\sigma_4\rangle(N, M_1)$ and (N, M_1) $[\sigma_5\rangle(N, M_2)[\sigma_2\rangle(N, M_o)$, such that σ_4 contains $\alpha_\lhd$. Since $\alpha_\lhd \gg \beta_\lhd$, firing sequence σ_1 or σ_5 must contain $\beta_\lhd$. As in the previous case, it follows that any firing sequence σ_3 between two markings $M_3, M_4 \in [N, M_i\rangle$ with $M_3(\kappa_\lhd) > 1$ and $M_4(\kappa_\rhd) > 1$ must contain $\beta_\lhd$.

(III) If $b(\gamma) = B_\diamond$, then $\epsilon_\lhd$ and $\kappa_\lhd$ are places in the post-set of transition $\gamma_\lhd$ (γ is a transition bordered bond). Let $M_5, M_6, M_7 \in [N, M_i\rangle$ be markings with $M_5(\kappa_\lhd) > 1$, $M_5(\epsilon_\lhd) > 1$, $M_6(\kappa_\lhd) > 1$, $M_6(\epsilon_\rhd) > 1$, $M_7(\kappa_\rhd) > 1$, $M_7(\epsilon_\rhd) > 1$. Then any firing sequence from M_5 to M_6 might contain $\alpha_\lhd$. Since $\alpha_\lhd \gg \beta_\lhd$, again, all firing sequences from M_6 to M_7 must contain $\beta_\lhd$.

For all three possible types of fragments for γ, we summarise that we have to ensure that any firing sequence leading from a marking that marks $\kappa_\lhd$ to a marking that marks $\kappa_\rhd$ must contain transition $\beta_\lhd$. Thus, for $\kappa_\lhd$ being a place, we know that $b(\kappa) \neq B_o$, and $b(\kappa) = L$ implies that $\forall\, \vartheta \in \Theta\, [\, \beta \in \chi^+(\vartheta)\,]$ with $\Theta = \{\vartheta \in \chi(\kappa) \mid \ell(\vartheta) = \Rightarrow\}$, cf., the argument for the very first case (2), if ϵ would be an arbitrary child of κ.

For both cases, $\kappa_\triangleleft$ being a transition or a place, fragment κ does not satisfy the assumptions on a fragment $\omega \in (\Pi \setminus \{\beta\})$. As this argument can be applied to all fragments on the path $\pi_{\mathcal{T}}(\kappa, \eta)$, we arrived at a contradiction with our assumption.

$\Leftarrow$ Let $\forall\ \omega \in (\Pi \setminus \{\beta\})$ either $t(\omega) = P$ or $b(\omega) = B_\diamond$, or if $b(\omega) = L$ then $\forall\ \vartheta \in (\chi(\omega) \cap \Pi)\ [\ \ell(\vartheta) \Longrightarrow\ ]$ with $\Theta = \{\vartheta \in \chi(\omega) \mid \ell(\vartheta) \Longrightarrow\}$. Assume $\alpha_\triangleleft \not\gg \beta_\triangleleft$. With δ as defined before, one path $\pi_N(\delta_\triangleleft, \beta_\triangleleft)$ is forwards conflict-free, i. e., $\delta_\triangleleft \gg \beta_\triangleleft$, by Lemma 5.1.7. For fragment η, we distinguish two cases.

(1) $\eta_\triangleleft$ is a transition. Then, $t(\eta) \in \{P, B\}$, and $t(\eta) = B$ requires $b(\eta) = B_\diamond$. Both imply that one path $\pi_N(\eta_\triangleleft, \delta_\triangleleft)$ is forwards conflict-free, i. e., $\eta_\triangleleft \gg \delta_\triangleleft$. With $\delta_\triangleleft \gg \beta_\triangleleft$ we also get $\eta_\triangleleft \gg \beta_\triangleleft$.

(2) $\eta_\triangleleft$ is a place. Then, $t(\eta) \in \{P, B\}$, and $t(\eta) = B$ requires $b(\eta) = L$. For $t(\eta) = P$, we get $t \gg \delta_\triangleleft$ for all transitions $t \in {}^\bullet\eta_\triangleleft$. For $t(\eta) = B$, we have $b(\eta) = L$ and $\forall\ \vartheta \in \Theta\ [\ \beta \in \chi^+(\vartheta)\ ]$ with $\Theta = \{\vartheta \in \chi(\eta) \mid \ell(\vartheta) \Longrightarrow\}$. As only one child of fragment η can contain fragment β, i. e., fragment δ, we know $|\Theta| = 1$. That is, there is only one path from $\eta_\triangleleft$ to $\eta_\triangleright$, represented by fragment δ. Therefore, $t \gg \delta_\triangleleft$ for all transitions $t \in {}^\bullet\eta_\triangleleft$. For both cases, $t(\eta) = P$ or $t(\eta) = B$, it also holds $t \gg \beta_\triangleleft$ for all transitions $t \in {}^\bullet\eta_\triangleleft$, since $\delta_\triangleleft \gg \beta_\triangleleft$.

We summarize that for both cases, we derive either $\eta_\triangleleft \gg \beta_\triangleleft$, or $t \gg \beta_\triangleleft$ for all transitions $t \in {}^\bullet\eta_\triangleleft$, respectively. Applying this argument to all fragments on the path $\pi_{\mathcal{T}}(\gamma, \eta)$ yields $\gamma_\triangleleft \gg \beta_\triangleleft$ or $t \gg \beta_\triangleleft$ for all transitions $t \in {}^\bullet\gamma_\triangleleft$, respectively. Trivially, $\alpha_\triangleleft \gg \gamma_\triangleleft$ if $\gamma_\triangleleft$ is a transition or $\alpha_\triangleleft \gg t$ for all transitions $t \in {}^\bullet\gamma_\triangleleft$ if $\gamma_\triangleleft$ is a place, due to γ being an ancestor of α. Thus, it holds $\alpha_\triangleleft \gg \beta_\triangleleft$, which is a contradiction with our assumption.

□

We illustrate both results using our example from Figure 28. For instance, transitions B and E are in strict order, $B \rightsquigarrow E$, as the LCA of the trivial fragments that have B and E as entries is the polygon fragment P2, cf., Figure 30. Here, the `order` value for the child fragment of P2 containing B is lower than the one for the child fragment that contains E. Further, the path from the root of the tree P1 to P2, i. e., $\pi_{\mathcal{T}}(P1, P2)$, does not contain any loop fragment. It holds $D + E$ for transitions D and E. Their LCA is the fragment B3 in Figure 28 or $B_\circ 1$ in Figure 30, respectively. The fragment $B_\circ 1$ is a place bordered bond and, again, the path $\pi_{\mathcal{T}}(P1, B_\circ 1)$ does not contain any loop fragment. Transitions B and C, in turn, are an example for interleaving order, $B \| C$, as their LCA is fragment B2 in Figure 28. This fragment corresponds to the loop type fragment L1 in Figure 30. Derivation of the co-occurrence is illustrated using transitions B and C. We see

that the path from the respective LCA (i. e., B2 in Figure 28, L1 in Figure 30) to the trivial fragments having B and C as entries contains solely polygon fragments, P4 and P5, respectively. However, the LCA itself is a loop fragment, such that the orientation of its child fragments P4 and P5 needs to be considered. There is only one child with forward orientation, namely P4. It contains transition B. Therefore, we derive $C \gg B$, but $B \not\gg C$ according to Theorem 5.2.5.

With these results, computation of the causal behavioural profile for a pair of transitions in a sound free-choice WF-system is done efficiently in the absence of rigid fragments.

Corollary 5.2.6. *The following problem can be solved in linear time. Given a sound free-choice WF-system* (N, M_i) *and its WF-tree* $\mathcal{T}_N$*, to compute the causal behavioural profile for a pair of transitions* (a, b) *if* b *is not contained in any rigid fragment.*

Proof. Let a and b be transitions and β be a trivial fragment of $\mathcal{T}_N$ with $b = \beta_\triangleleft$. Computation of the relations according to Theorem 5.2.4 and Theorem 5.2.5 requires analysis of fragments on a subpath from the root of $\mathcal{T}_N$ to β. The analysis of a single fragment is performed in constant time. In the worst case, the length of the subpath is linear in size to the number of fragments in $\mathcal{T}_N$. The number of fragments in $\mathcal{T}_N$ is linear to the number of flows in the WF-system [37, 190, 359]. □

Complete Computation Algorithm

We integrate the results for the computation of causal behavioural profiles based on structural decomposition with those obtained in Section 5.1 in a comprehensive algorithm. The algorithm expects a sound free-choice WF-system and a pair of transitions as input. Given the input, the algorithm determines the profile relation and checks the co-occurrence relation for the pair of transitions. Besides the already presented theory, the algorithm exploits the following result. It defines a link between interleaving order for a pair of transitions and the existence of a cyclic path that contains both transitions.

Lemma 5.2.7. *Let* (N, M_i), $N = (P, T, F)$*, be a sound free-choice WF-system,* $\mathcal{T}_N = (\Omega, \chi, t, b)$ *the WF-tree of* N*, and* $\omega = (P', T', F')$, $\omega \in \Omega$*, a fragment of* N*. If there exists a path* $\pi_N(\omega_\triangleright, \omega_\triangleleft)$*, then* $t_1 \| t_2$*, for all transitions* $t_1, t_2 \in T'$*.*

Proof. The existence of a path $\pi_N(x, y)$, where $x, y \in T \cup P$, implies the existence of a firing sequence containing all transitions on $\pi_N(x, y)$ due to free-choiceness and soundness [234]. The claim immediately follows from the fact that there are two paths, i. e., $\pi_N(\omega_\triangleright, \omega_\triangleleft)$ and $\pi_N(\omega_\triangleleft, \omega_\triangleright)$. □

Finally, Algorithm 1 details the steps to take when computing the relations of the causal behavioural profile for a pair of transitions.

Algorithm 1: Computation of the causal behavioural profile for a transition pair.

Input: A sound free-choice WF-system (N, M_i) with $N = (P, T, F)$ and two transitions $x, y \in T$.
Output: Causal behavioural profile relations for x and y.

1 $\mathcal{T}_N = (\Omega, \chi, t, b)$, the WF-tree of N;
2 $\alpha \in \Omega$, a trivial fragment with entry x;
3 $\beta \in \Omega$, a trivial fragment with entry y;
4 $\omega_r \in \Omega$, the root fragment of $\mathcal{T}_N$;
5 $\gamma = \mathrm{lca}(\alpha, \beta)$;

`// Compute profile relation`
6 **if** $\forall\ \omega \in \pi_{\mathcal{T}}\{\omega_r, \gamma\}\ [\ t(\omega) \neq R\ ]$ **then**
7 | Get profile relation for x and y using Theorem 5.2.4;
8 **else**
9 | Check interleaving order for x and y by Lemma 5.2.7 on subnet γ;
10 | **if** *not* $x \parallel y$, *according to Lemma 5.2.7* **then**
11 | | Perform state space exploration for subnet γ to derive profile relation for x and y;
12 | **end**
13 **end**

`// Check co-occurrence relation`
14 **if** $\forall\ \omega \in \pi_{\mathcal{T}}\{\gamma, \beta\}\ [\ t(\omega) \neq R\ ]$ **then**
15 | Check co-occurrence relation for x and y by Theorem 5.2.5;
16 **else**
17 | **if** γ *is a T-net* **then**
18 | | Check co-occurrence for x and y by Theorem 5.1.8;
19 | **else if** γ *is an S-net* **then**
20 | | Check co-occurrence for x and y by Theorem 5.1.9;
21 | **else if** γ *is acyclic* **then**
22 | | Check co-occurrence for x and y by Theorem 5.1.10;
23 | **else**
24 | | Perform state space exploration to check co-occurrence for x and y;
25 | **end**
26 **end**

Algorithm 1 comprises three stages. First, required data structures are initialized (lines 1 to 5). Second, computation of the behavioural profile relation for the given pair of transitions takes place (lines 6 to 13). Last, the pair of transitions is checked for being in the co-occurrence relation (lines 14 to 26). If there exists no rigid fragment on the path from the root of the WF-tree to fragment γ, the profile relation is derived using Theorem 5.2.4 (line 7); otherwise, the algorithm checks if the given transitions are in interleaving order by Lemma 5.2.7 (line 9), or computes

the relations for the whole fragment γ by state space exploration and extracts the requested relation (line 11). As the profile relations are grounded on trace semantics, exploration of all traces would be sufficient to derive the respective relation. Still, the set of traces may be infinite. Therefore, we rely on the exploration of the state space instead. Soundness of the WF-system guarantees that the state space is finite. The check of the co-occurrence relation, in the absence of rigid fragments in the WF-tree on the path from γ to β, relies on Theorem 5.2.5; otherwise the checks depend on the structural class of fragment γ (lines 17 to 22). Lastly, if γ does not meet the structural assumptions, again, state space exploration is needed to decide whether the transition pair is co-occurring (line 24).

5.3 COMPUTATIONS FOR BOUNDED SYSTEMS

The methods for the computation of behavioural profiles introduced in the previous sections impose structural and behavioural assumptions on net systems. In Section 2.3, we discussed that these assumptions hold for a broad class of net systems that are derived from models captured in common process description languages. Still, we also presented some evidence that these assumptions cannot be assumed to hold in all cases. Therefore, this section introduces a more generic approach to the computation of behavioural profiles. It is applicable for all net systems that are bounded. The behaviour of these systems is characterised by a finite set of states, i. e., markings. Any analysis of this set of states has to cope with the state explosion problem [444]. Our approach leverages the notion of a complete prefix unfolding, which has been proposed to address this problem [305, 149]. The generality of the proposed approach is traded for computational complexity. The computation of behavioural profiles for bounded systems is computationally harder than the techniques introduced in the previous sections. The construction of a complete prefix unfolding, the preliminary step for our approach, is known to be an NP-complete problem [204, 149].

In the remainder of this section, we first present basic definitions for complete prefix unfoldings. Second, we establish the relation between the behavioural profile and the complete prefix unfolding, and present an algorithm for the derivation of the behavioural profile. Finally, we focus on the causal behavioural profile and introduce an algorithm for the computation of the co-occurrence relation.

Complete Prefix Unfoldings

The unfolding of a net system is another, potentially infinite net system, which has a simpler, tree-like structure [305, 149]. We recall definitions for unfoldings based on [151].

Definition 5.3.1 (Occurrence Net, Order Relations)
Let $N = (P, T, F)$ be a net, $X = (P \cup T)$ its set of nodes, and F^+ (F^*) the irreflexive (reflexive) transitive closure of F.

- A pair of nodes $(x, y) \in (X \times X)$ is in the *conflict relation* $\#$, iff $\exists\ t_1, t_2 \in T\ [\ (t_1 \neq t_2) \wedge (\bullet t_1 \cap \bullet t_2 \neq \emptyset) \wedge (t_1\ F^*\ x) \wedge (t_2\ F^*\ y)\ ]$.
- N is an *occurrence net*, iff (1) N is acyclic, (2) $\forall\ p \in P\ [\ |\bullet p| \leqslant 1\ ]$, and (3) for all $x \in X$ it holds $x \# x$ and the set $\{y \in X \mid y\ F^+\ x\}$ is finite. In an occurrence net, transitions are called *events*, and places are called *conditions*.
- If N is an occurrence net, the relation F^+ is the *causality relation* and denoted by $<$, F^* is denoted by $\leqslant$. A pair of nodes $(x, y) \in (X \times X)$ of N is in the *concurrency relation* co, if neither $x \leqslant y$, nor $y \leqslant x$, nor $x \# y$.
- If N is an occurrence net, $Min(N)$ denotes the set of minimal elements of X w.r.t. $\leqslant$.

The relation between a net system $S = (N, M_0)$ with $N = (P, T, F)$ and an occurrence net $O = (C, E, G)$ is defined as a homomorphism $h : (C \cup E) \mapsto (P \cup T)$ such that $h(C) \subseteq P$ and $h(E) \subseteq T$; for all $e \in E$, the restriction of h to $\bullet e$ is a bijection between $\bullet e$ and $\bullet h(e)$; the restriction of h to $e\bullet$ is a bijection between $e\bullet$ and $h(e)\bullet$, the restriction of h to $Min(O)$ is a bijection between $Min(O)$ and all places marked in M_0; and for all $e, f \in E$, if $\bullet e = \bullet f$ and $h(e) = h(f)$ then $e = f$.

A *branching process* of $S = (N, M_0)$ is a tuple $\pi = (O, h)$ with $O = (C, E, G)$ being an occurrence net and h being a homomorphism from O to S as defined before. A branching process $\pi' = (O', h')$ is a *prefix*, if $O' = (C', E', G')$ is a subnet of O, such that if $e \in E'$ and $(c, e) \in G$ or $(e, c) \in G$ then $c \in C'$; if $c \in C'$ and $(e, c) \in G$ then $e \in E'$; h' is the restriction of h to $C' \cup E'$.

The maximal branching process of S is called *unfolding*. The unfolding of a net system can be truncated once all markings of the original net system and all enabled transitions are represented. This yields the complete prefix unfolding.

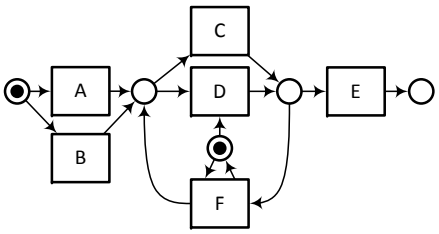
Definition 5.3.2 (Complete Prefix Unfolding)
Let $S = (N, M_0)$ be a system and $\pi = (O, h)$ a branching process with $N = (P, T, F)$, $X = (P \cup T)$, and $O = (C, E, G)$.

- A set of events $E' \subseteq E$ is a *configuration*, iff $\forall\ e, f \in E'\ [\ e \# f\ ]$ and $\forall\ e \in E'\ [\ f < e \Rightarrow f \in E'\ ]$. The *local configuration* $\lceil e \rceil$ for an event $e \in E$ is defined as $\{x \in X \mid x < e\}$.

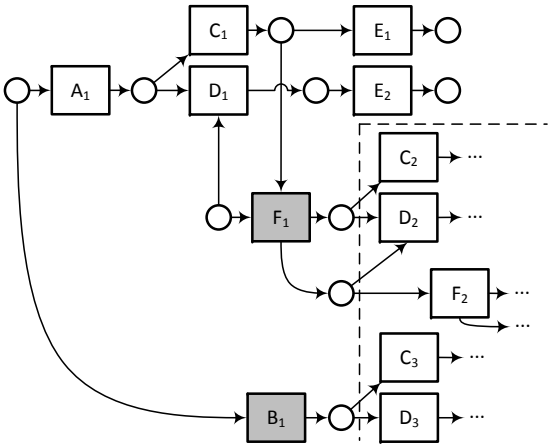
- A set of conditions $C' \subseteq C$ is called *co-set*, iff for all distinct conditions $c_1, c_2 \in C'$ it holds c_1 *co* c_2. If C' is maximal w.r.t. set inclusion, then it is called a *cut*.
- For a finite configuration C', $Cut(C') = (Min(O) \cup C'\bullet) \setminus \bullet C'$ is a cut, and $h(Cut(C'))$ is a marking of S reachable from M_0, denoted by $Mark(C')$.
- The branching process is *complete*, iff for every marking $M \in [N, M_0\rangle$ there is a configuration C' of π such that $M = Mark(C')$ and for every transition t enabled in M there is a finite configuration C' and an event $e \notin C'$ such that $M = Mark(C')$, $h(e) = t$, and $C' \cup \{e\}$ is a configuration.
- An *adequate order* $\lhd$ is a well-founded partial order on finite configurations such that (1) for two configurations C', C'' of π it holds that $C' \subset C''$ implies $C' \lhd C''$ and (2) $\lhd$ is preserved by finite extensions, i. e., if $C' \lhd C''$ and $Mark(C') = Mark(C'')$ then the futures of C' and C'', the suffixes of π beginning with $Cut(C')$ and $Cut(C'')$, are isomorphic.
- An event $e \in E$ is a *cut-off event* induced by $\lhd$, iff there is a *corresponding event* $f \in E$ with $Mark(\lceil e \rceil) = Mark(\lceil f \rceil)$ and $\lceil f \rceil \lhd \lceil e \rceil$.
- The branching process π is the *complete prefix unfolding* induced by $\lhd$, iff it is the greatest prefix of the unfolding of S that does not contain any event after a cut-off event.

The definition of a cut-off event and, therefore, of the complete prefix unfolding is parameterised by the definition of an adequate order $\lhd$. Multiple definitions have been proposed in the literature [149]. The differences between these definitions can be neglected for our approach. For the implementation and experimental evaluation, we rely on the definition presented in [151]. As we leverage the information on cut-off events in our approach, we include them in the complete prefix for convenience.

Figure 32 illustrates the concept of an unfolding and its complete prefix. Figure 32a depicts an example net system. This system is not free-choice, so that the approaches to the computation of causal behavioural profiles introduced in Section 5.1 and Section 5.2 are not applicable. Figure 32b depicts a part of the unfolding of the net system. Here, the labelling of transitions hints at the homomorphism between the two systems. For instance, both events C_1 and C_2 in the prefix in Figure 32b relate to the transition C of the original system in Figure 32a. The unfolding is infinite, which is caused by the circuit in the original net system. In Figure 32b, cut-off events are highlighted in grey and the complete prefix unfolding is marked by dashed lines.



(a) Net system that is not free-choice.



(b) The complete prefix unfolding of (a).

Figure 32: A net system and its complete prefix unfolding.

Derivation of the Profile Relations

We derive the relations of the behavioural profile directly from the relations of the complete prefix unfolding that have been introduced for occurrence nets in Definition 5.3.1. The causality, conflict, and concurrency relation partition the Cartesian product of events of the complete prefix unfolding. Although this resembles the partitioning induced by the profile relations, the relations of an occurrence net relate to *events*, i. e., *occurrences* of transitions of the original net system.

We deduce the weak order relation from the concurrency and the causality relation of the complete prefix unfolding. The existence of a firing sequence containing two transitions of the original system is reflected by two events in the prefix that relate to these transitions and are concurrent or in causality. The former represents two transitions that can be enabled concurrently in the original system, such that there is a firing sequence containing both transitions in either order. Two events in causality in the prefix, in turn, represent two transitions in the original net that can occur in a firing sequence in the respective order.

Still, not all firing sequences are visible in the complete prefix unfolding directly. Events that relate to two transitions may not show causality or concurrency although the respective trans-

itions may occur in a firing sequence. Consider, for instance, transitions B and E of the system in Figure 32a. Even though both transitions may be observed in some firing sequence, the pairs of corresponding events B_1 and E_1 (B_1 and E_2, respectively) are in conflict in the complete prefix unfolding in Figure 32b. In Figure 32b, the cuts of the local configurations of the events A_1 and B_1 represent the same marking in the original net system. This has to be taken into account when checking for weak order, i. e., the existence of a firing sequence in the original net system. We address this issue by the notion of an event continuation relation for the complete prefix unfolding. It holds between two events, if they are directly related by concurrency or causality, or if there is an indirect continuation between them through cut-off and corresponding events.

Definition 5.3.3 (Event Continuation)
Let $S = (N, M_0)$ be a bounded system and $\pi = (O, h)$ its complete prefix unfolding including cut-off events with $N = (P, T, F)$ and $O = (C, E, G)$. A pair of events $(e_1, e_2) \in (E \times E)$ is in the *event continuation relation* $\hookrightarrow$, iff

- they are distinct and either causally related or concurrent, $(e \neq f) \wedge ((e < f) \vee (e \mathrel{co} f))$, or
- there is a sequence of cut-off events $(g_1, \ldots, g_n)$ with $g_i \in E$ for $1 \leqslant i \leqslant n$ and a sequence of corresponding events $(g'_1, \ldots, g'_n)$ with $g'_i \in E$, such that all of the following requirements are met:
  - $e \leqslant g_1$,
  - $g'_j \leqslant g_{j+1}$ for $1 \leqslant j < n$,
  - $(g'_n \neq f) \wedge ((g'_n < f) \vee (g'_n \mathrel{co} f))$.

The existence of a continuation between two events in the complete prefix unfolding coincides with the existence of a firing sequence that comprises the transitions in the respective order in the original net system. Hence, an event continuation allows for concluding on weak order between the transitions.

Theorem 5.3.1. *Let $S = (N, M_0)$ be a bounded system and $\pi = (O, h)$ its complete prefix unfolding including cut-off events with $N = (P, T, F)$ and $O = (C, E, G)$. Then, two transitions $x, y \in T$ are in weak order, $x \succ y$, iff there are events $e, f \in E$ with $h(e) = x$ and $h(f) = y$, such that $e \hookrightarrow f$.*

Proof. Let $S = (N, M_0)$ and $\pi = (O, h)$ be defined as before and let $x, y \in T$ be transitions.

$\Rightarrow$ Let $x \succ y$. Then, there is a firing sequence in S starting in M_0 that contains transition x before y. As the prefix is complete, both occurrences of transitions are represented by corresponding events $e, f \in E$ with $h(e) = x$ and $h(f) = y$. If there is a configuration in O that contains e

before f, they are distinct and causally related or concurrent, i. e., $(e \neq f) \wedge ((e < f) \vee (e \mathit{co} f))$ (first statement of the definition of an event continuation, Definition 5.3.3). If there is no such configuration, then either $e \# f$ or $f < e$. If there is no cut-off event $k \in E$ in O with $e < k$, then the event f cannot refer to the firing of transition y as the events E and f either represent occurrences of transitions that cannot occur together ($e \# f$) or that may occur only in the reversed order ($f < e$). Therefore, there has to be a cut-off event k with either $e \leqslant k$ or $e \mathit{co} k$. Assume that the sequence of cut-off events is one. Then, to explain the occurrence of transition y after transition x in a firing sequence in S, it holds that the corresponding event k' for k is related to f by either $k' < f$, $f < k'$, or $f \mathit{co} k'$. Coming back to the two cases, $e \leqslant k$ or $e \mathit{co} k$, we first consider the latter. If $e \mathit{co} k$, then the occurrence of transition x represented by e happens only after the marking represented by the cut of $\lceil k \rceil$ is reached. To explain an occurrence of transition x before transition z, thus, there has to be an event e' with $h(e) = h(e')$ and $e' \mathit{co} k'$ for which holds $(e' \neq f) \wedge ((e' < f) \vee (e' \mathit{co} f))$. This, again, is covered by the definition of an event continuation. Consider $e \leqslant k$. Then, to explain the occurrence of transition y after transition x in a firing sequence in S, it holds that the corresponding event k' for k is distinct ($k' \neq f$) to f and either $k' < f$, $f < k'$, or $f \mathit{co} k'$. To observe a firing of x before y, we have to exclude $f < k'$ from the possible relations between f and k'. That is because $f < k'$ implies that y is observed before the marking represented by the cut of $\lceil k \rceil$ is reached as $\mathit{Mark}(\lceil k \rceil) = \mathit{Mark}(\lceil k' \rceil)$. Hence, we have $k' < f$ or $k' \mathit{co} f$, which is the second statement of the definition of an event continuation, Definition 5.3.3. The same argument is applied to all intermediate cut-off events in case the sequence of cut-off events is longer than one. Here, the requirements for the relation between events e and k ($e \leqslant k$) are enforced to all intermediate events ($g'_j \leqslant g_{j+1}$).

$\Leftarrow$ For all events $e, f \in E$ that meet $h(e) = x$ and $h(f) = y$ let $e \hookrightarrow f$. The latter translates into (1) $(e \neq f) \wedge ((e < f) \vee (e \mathit{co} f))$, or (2) there is a sequence of cut-off events $(g_1, \ldots, g_n)$ and corresponding events $(g'_1, \ldots, g'_n)$ that satisfies the requirements given in Definition 5.3.3. Assume that $x \not\succ y$. Let M_1 be the marking in S represented by the cut $\mathit{Cut}(\lceil f \rceil)$. If $e < f$ then event e is part of $\lceil f \rceil$. Hence, transition x has been fired to reach marking M_1 in S, which is a contradiction with $x \not\succ y$. If $e \mathit{co} f$ and $e \neq f$, there must be an event g in $\lceil f \rceil$ with $g < e$ and for every condition $c_f \in {\bullet f}$ there is a condition $c_g \in g{\bullet}$ with $c_g < c_f$ or $c_g = c_f$. Since $e \mathit{co} f$, $e \neq f$, $g < f$, and $g < e$, we also know $c_g \not< e$

for all those conditions c_g. Hence, these conditions c_g are part of the cut $Cut(\lceil e \rceil)$. Let M_2 be the marking in S that is represented by this cut. There is a firing sequence starting in M_2 that comprises all transitions that are represented by the events $\lceil f \rceil \setminus \lceil e \rceil$. Hence, there is a firing sequence starting in M_2 that contains transition y. This is a contradiction with $x \not\succ y$. Consider case (2). Following on the argument given in the previous case, we know that there is a firing sequence in S that reaches a marking M_3 and comprises the transition $h(e)$, i. e., transition x, before the transition $h(g_1)$, if $e \neq g_1$. For the corresponding event g'_1, we know $Mark(\lceil g_1 \rceil) = Mark(\lceil g'_1 \rceil)$. Hence, M_3 is also reached in S through firing all transitions represented by events in $\lceil g'_1 \rceil$. Assume that the sequence of cut-off events is one, i. e., $(g'_1 \neq f) \wedge ((g'_1 < f) \vee (g'_1 \; co \; f))$. Then, there is a firing sequence in S starting in M_3 that contains the transition $h(g'_1)$ before the transition $h(f)$, i. e., transition y. As M_3 may be reached by a firing sequence containing transition x, there is a firing sequence comprising transition x before transition y. We arrived at a contradiction with $x \not\succ y$. The same argument is applied to all intermediate cut-off events in case the sequence of cut-off events is longer than one.

□

Algorithm 2 shows how the behavioural profile is computed from the complete prefix unfolding of a bounded system.

First, we compute the order relations, i. e., the causality, conflict, and concurrency relation, for the complete prefix unfolding (line 1). The respective algorithm can be found in [243].

Second, we identify cut-off and corresponding events (lines 2 to 7). The set $\mathcal{E}_{cut}$ is filled with all cut-off events, their corresponding events are added to the set $\mathcal{E}_{cor}$. We store the relation between them in $\mathcal{E}$.

Third, we build the event continuation relation for the complete prefix unfolding (lines 8 to 21). For each pair of a corresponding event of some cut-off event and another cut-off event, we check whether they are identical or causally related. If so, the relation between both events is also added to the relation $\mathcal{E}$. The intuition behind is that the transitive closure of $\mathcal{E}$ hints at the existence of a sequence of cut-off and corresponding events, cf., Definition 5.3.3.

Fourth, all pairs of events of the complete prefix unfolding are assessed for the existence of an event continuation between them (lines 22 to 25). If so, the weak order relation is captured for the transitions that are represented by these events according to Theorem 5.3.1.

Finally, the relations of the behavioural profile are derived from the weak order relation (lines 26 to 31).

Algorithm 2: Computation of the behavioural profile from the complete prefix unfolding.

Input: $S = (N, M_0)$, a bounded system with $N = (P, T, F)$.
$\pi = (O, h)$, its complete prefix unfolding including cut-off events with $O = (C, E, G)$.
Output: $\mathcal{B} = \{\rightsquigarrow, +, \|\}$, the behavioural profile of S.

1 Compute order relations $<$, $\#$, and co of O;

// Compute cut-off events and corresponding events in π
2 $\mathcal{E}_{cor}, \mathcal{E}_{cut}, \mathcal{E} \longleftarrow \emptyset$;
3 **foreach** $(e_1, e_2) \in (E \times E)$ **do**
4 | **if** $(Mark(\lceil e_1 \rceil) = Mark(\lceil e_2 \rceil)) \wedge (\lceil e_1 \rceil \lhd \lceil e_2 \rceil)$ **then**
5 | | $\mathcal{E}_{cor} \longleftarrow (e_1)$; $\mathcal{E}_{cut} \longleftarrow (e_2)$; $\mathcal{E} \longleftarrow (e_2, e_1)$;
6 | **end**
7 **end**

// Build event continuation relation of π
8 $\hookrightarrow \longleftarrow \emptyset$;
9 **foreach** $(e_{cor}, e_{cut}) \in (\mathcal{E}_{cor} \times \mathcal{E}_{cut})$ **do**
10 | **if** $e_{cor} \leqslant e_{cut}$ **then** $\mathcal{E} \longleftarrow (e_{cor}, e_{cut})$;
11 **end**
12 **foreach** $(e_1, e_2) \in (E \times E)$ **do**
13 | **if** $(e_1 \neq e_2) \wedge ((e_1 < e_2) \vee (e_1 \; co \; e_2))$ **then** $\hookrightarrow \longleftarrow (e_1, e_2)$;
14 | **else**
15 | | **foreach** $(e_{cut}, e_{cor}) \in (\mathcal{E}_{cut} \times \mathcal{E}_{cor})$ **do**
16 | | | **if** $(e_1 \leqslant e_{cut}) \wedge (e_{cut} \; \mathcal{E}^+ \; e_{cor}) \wedge$
$(e_{cor} \neq e_2) \wedge ((e_{cor} < e_2) \vee (e_{cor} \; co \; e_2))$ **then**
17 | | | | $\hookrightarrow \longleftarrow (e_1, e_2)$;
18 | | | **end**
19 | | **end**
20 | **end**
21 **end**

// Derive weak order for transitions of S
22 $\succ \longleftarrow \emptyset$;
23 **foreach** $(e_1, e_2) \in (E \times E)$ **do**
24 | **if** $(e_1 \hookrightarrow e_2)$ **then** $\succ \longleftarrow (h(e_1), h(e_2))$;
25 **end**

// Derive relations of behavioural profile of S
26 $\rightsquigarrow, +, \| \longleftarrow \emptyset$;
27 **foreach** $(t_1, t_2) \in (T \times T)$ **do**
28 | **if** $(t_1 \succ t_2) \wedge (t_2 \succ t_1)$ **then** $\| \longleftarrow (t_1, t_2)$;
29 | **else if** $t_1 \succ t_2$ **then** $\rightsquigarrow \longleftarrow (t_1, t_2)$;
30 | **else** $+ \longleftarrow (t_1, t_2)$;
31 **end**

Proposition 5.3.2. *Algorithm 2 terminates and after termination* $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ *is the behavioural profile of* S.

Proof. Termination: The algorithm iterates over sets that are derived from E, C, $\mathcal{E}_{cut} \subseteq E$, $\mathcal{E}_{cor} \subseteq E$, and T. T is finite by definition. Since the net system is bounded, the complete prefix unfolding and, therefore, the sets of events E and conditions C are finite as well. Hence, the algorithm terminates.
Result: Relation $\mathcal{E}$ is built such that it contains cut-off events and their corresponding events. Further, it contains events that are corresponding to a cut-off event along with all cut-off events that are identical or in causality to the former event. Hence, the transitive closure of $\mathcal{E}$ hints at the existence of a sequence of cut-off events and corresponding events that are related as stated in Definition 5.3.3. Further, the statements of Definition 5.3.3 are implemented directly to compute the event continuation relation. Derivation of the weak order relation is realised according to Theorem 5.3.1. Finally, derivation of the profile relations based on weak order follows directly on the definition of the behavioural profile. □

The algorithm runs in polynomial time with respect to the size of the complete prefix unfolding. The final step of the algorithm, which sets the profile relations based on the weak order relation for all pairs of transitions, is neglected at this point.

Corollary 5.3.3. *The following problem can be solved in* $O(n^4)$ *time with* n *as the number of events and conditions of the complete prefix unfolding:*
For a bounded net system and its complete prefix unfolding, to compute the weak order relation for the net system.

Proof. We assume all relations used and created in the algorithm to be represented by their characteristic functions, i. e., to be encoded as bi-dimensional arrays that map to either zero or one. Then, adding an entry to a relation and checking membership for a tuple takes constant time. Computation of the order relations of the complete prefix unfolding is done in $O(|E| \cdot |C|)$ time [243]. In the second step of the algorithm, we iterate over $E \times E$, which takes $O(|E|^2)$ time. In the third step, we iterate over $\mathcal{E}_{cut} \times \mathcal{E}_{cor}$. Since $\mathcal{E}_{cut} \subseteq E$ and $\mathcal{E}_{cor} \subseteq E$, this takes $O(|E|^2)$ time. Then, we iterate over $E \times E \times \mathcal{E}_{cut} \times \mathcal{E}_{cor}$, which takes $O(|E|^4)$ time. As a prerequisite for this step the transitive closure of $\mathcal{E}$ is computed, which takes $O(|E|^3)$ time [486]. The iteration over $E \times E$ to set weak order requires $O(|E|^2)$ time. Overall time complexity is $O(n^4)$ with n as the number of events and conditions of the complete prefix unfolding. □

The algorithm runs in polynomial time to the size of the complete prefix unfolding, not to the size of the net system. The

adequate order presented in [151] to parameterise the definition of a complete prefix unfolding has been shown to create compact prefixes. Nevertheless, the prefixes may be large, at most the size of the state space of the net system [151]. Hence, even a polynomial time algorithm may result in a high computational effort. Later, we will present experimental results to investigate this issue.

Derivation of the Co-occurrence Relation

The previous section established the relation between the existence of a firing sequence containing two transitions and the relations of the respective events in the complete prefix unfolding. It suffices to investigate the existence of a firing sequence to derive the relations of the behavioural profile. The computation of the co-occurrence relation, in turn, requires investigation of all possible firing sequences.

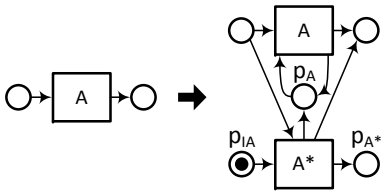


Figure 33: Pre-processing for unfolding.

We tackle the computation of co-occurrences by augmenting the original net system with additional transitions and places. This transformation aims at making co-occurrences directly visible in the complete prefix unfolding. The idea is to have a dedicated place signalling that a certain transition has been fired at least once. Adding a place to the post-set of a transition, however, may result in an unbounded net system. Any transition that may be fired infinitely often, e. g., as part of a circuit, would cause an infinite number of reachable markings. Therefore, we apply the transformation illustrated in Figure 33 for all transitions for which co-occurrence should be determined. For the transition A, an additional transition A^* is inserted. Informally, this transition represents the first firing of transition A in the original system. After transition A^* has been fired, two places, p_A and p_{A^*}, are marked. The former place ensures that transition A can only be fired after the inserted transition A^* has been fired before. The latter place remains marked to signal that transition A^* has been fired. This, in turn, indicates that transition A in the original system has been fired at least once. We define the transformation that yields an augmented net system as follows.

Definition 5.3.4 (Augmented Net System)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$ and $t \in T$ a transition.

- The *augmented net system* of S induced by t is a net system $\hat{S} = (\hat{N}, \hat{M}_0)$ with $\hat{N} = (\hat{P}, \hat{T}, \hat{F})$, such that
  - $\hat{T} = T \cup \{t^*\}$ with t^* being a fresh transition, $t^* \notin T$, t^* is called *augmented transition* for t,
  - $\hat{P} = P \cup \{p_{it}, p_t, p_{t^*}\}$ with p_{it}, p_t, and p_{t^*} being three fresh places, $p_{it}, p_t, p_{t^*} \notin P$, p_{t^*} is called *augmented place* for t,
  - $\hat{F} = F \cup \{(p, t^*) \mid (p, t) \in F\} \cup \{(t^*, p) \mid (t, p) \in F\} \cup \{(p_{it}, t^*), (t^*, p_t), (t^*, p_{t^*}), (t, p_t), (p_t, t)\}$,
  - $\hat{M}_0 = M_0 \cup (p_{it}, 1)$,
- For the augmented system $\hat{S}$ of S induced by t, and t^* and p_{t^*} as the augmented transition and place, the *augmentation function* $a : (P \cup T) \to T$ is defined as

$$a(n) = \begin{cases} t, & \text{iff } n = t^* \ \vee \ n = p_{t^*} \\ \bot, & \text{else.} \end{cases}$$

- The augmented net system of S induced by $T' \subseteq T$ is derived by a step-wise augmentation with all transitions $t' \in T'$.

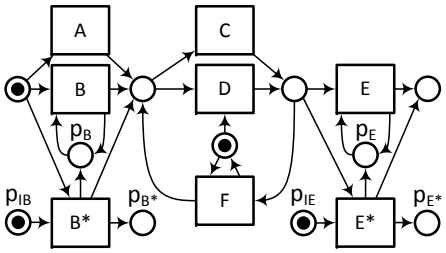
For a net system $S = (N, M_0)$, $N = (P, T, F)$, we refer to the augmented net system of S induced by all transitions T as the augmented system of S. We exemplify the augmentation of net systems with the example net introduced in Figure 32a. Figure 34a shows the augmented net system that is induced by two transitions, B and E. In this system, the places p_{B^*} and p_{E^*} indicate that transitions B and E in the original net system in Figure 32a have been fired. There is a close relation between the behaviour of a system and the behaviour of the augmented system. To show this relation, we introduce the notion of a corresponding firing sequence.

Definition 5.3.5 (Corresponding Firing Sequence)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$. Let $\hat{S} = (\hat{N}, \hat{M}_0)$ be the augmented net system with $\hat{N} = (\hat{P}, \hat{T}, \hat{F})$ and a the respective augmentation function. A firing sequence $\sigma = \langle t_1, \ldots, t_n \rangle$ in S, $(N, M_0)[\sigma\rangle$, *corresponds* to a firing sequence $\hat{\sigma} = \langle s_1, \ldots, s_n \rangle$ in $\hat{S}$, $(\hat{N}, \hat{M}_0)[\hat{\sigma}\rangle$, iff with $1 \leqslant i \leqslant n$ it holds

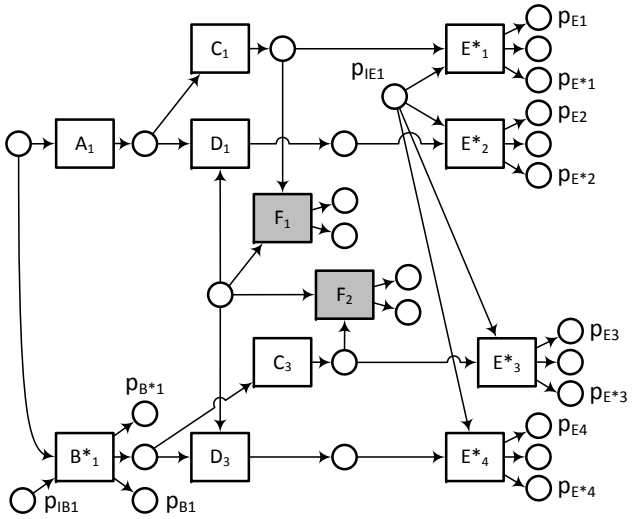
- $t_i = a(s_i)$, if $\forall\, j \in \mathbb{N}, j < i \,[\, t_j \neq t_i \,]$, and
- $t_i = s_i$, otherwise.

In fact, for each firing sequence in either system, there exists a corresponding firing sequence.

Lemma 5.3.4. *The relation between corresponding firing sequences of a net system and of the augmented net system is a bijection.*



(a) The augmented net system of the net system in Figure 32a induced by transitions {B, E}.



(b) The complete prefix unfolding of (a).

Figure 34: An augmented net system and its complete prefix unfolding

Proof. Let $S = (N, M_0)$ be a net system, $N = (P, T, F)$, and $\hat{S} = (\hat{N}, \hat{M}_0)$ the augmented net system, $\hat{N} = (\hat{P}, \hat{T}, \hat{F})$. Let $t \in T$ be a transition. For this transition, the augmentation step adds three places, p_{it}, p_t, and p_{t^*}, and one transition, t^*, to the original net structure. Place p_{it} has an empty pre-set and is marked initially. Transition t^* is the only transition that consumes a token from p_{it}. Hence, transition t^* fires at most once as part of a firing sequence that starts in the initial marking. Transition t, in turn, requires place p_t to be marked, which is the case only once transition t^* has been fired. As all places in the pre- and post-set of t are also in the pre- and post-set of t^*, every marking that enables t in S enables either t or t^* in $\hat{S}$, depending on whether transition t^* was fired to reach the respective marking. Therefore, any firing sequence in S is rewritten such that the first occurrence of transition t is replaced by an occurrence of t^* to yield a firing sequence in the augmented net system $\hat{S}$. For any firing sequence in $\hat{S}$, the occurrence of transition t^* is replaced by an occurrence of t to yield a firing sequence in S. This argument holds for all transitions $t \in T$. □

We established the relation between a system and the augmented system by means of corresponding firing sequences. Corresponding firing sequences imply a relation between the markings reached by firing them in both systems. We refer to these markings as corresponding markings.

We motivated the definition of the augmentation step by the need to preserve boundedness of net systems. This property is not affected by the transformation, indeed.

Lemma 5.3.5. *For any bounded net system holds, the augmented net system is bounded.*

Proof. Let $S = (N, M_0)$ be a net system, $N = (P, T, F)$, and $\hat{S} = (\hat{N}, \hat{M}_0)$ the augmented net system, $\hat{N} = (\hat{P}, \hat{T}, \hat{F})$ induced by a transition $t \in T$. The augmentation step adds three places, p_{it}, p_t, and p_{t^*}, to the original net structure. Place p_{it} has an empty pre-set and one transition, t^*, in its post-set. Place p_{it} is marked in $\hat{M}_0$, so that transition t^* can fire at most once. Firing of transition t^* marks place p_{t^*}, which remains marked as it has an empty post-set. Transition t^* is the only transition that puts a token into place p_t without consuming a token from this place. Place p_t is not marked initially. Hence, in every reachable marking $\hat{M} \in [\hat{N}, \hat{M}_0\rangle$ either place p_{it} is marked or places p_t and p_{t^*} are marked. All three places can be marked with at most one token. Thus, any marking reachable in S has at most two reachable corresponding markings in $\hat{S}$ that mark either place p_{it} or places p_t and p_{t^*}; and are equal for all places $p \in P$. Hence, augmentation induced by one transition at most duplicates the number of reachable markings in the augmented system compared to the original system. Since, the latter is bounded and the set of transitions exploited for augmentation is finite, the number of reachable markings in the augmented system is finite. □

Using the notion of an augmented net system, we decide co-occurrence for two transitions of a bounded net system.

Theorem 5.3.6. *Let $S = (N, M_0)$ be a bounded system with $N = (P, T, F)$ and $x, y \in T$ two transitions. Let $\hat{S} = (\hat{N}, \hat{M}_0)$ with $\hat{N} = (\hat{P}, \hat{T}, \hat{F})$ be the augmented net system of S induced by $\{x, y\}$ with a as the augmentation function. Let $\pi = (O, h)$ be the complete prefix unfolding including cut-off events of $\hat{S}$ with $O = (C, E, G)$. Then, x and y are co-occurring, $x \gg y$, iff for all events $e \in E$ with $\exists\, c \in Cut(\lceil e \rceil)\ [\, a(h(c)) = x\,]$ it holds either*

- *there is a condition $c \in Cut(\lceil e \rceil)$ with $a(h(c)) = y$, or*
- *there is an event $f \in E$ with $a(h(f)) = y$ and $(e = f) \vee (e \hookrightarrow f)$.*

Proof. Let $S = (N, M_0)$, $\hat{S} = (\hat{N}, \hat{M}_0)$, and $\pi = (O, h)$ be defined as before and let $x, y \in T$ be transitions.

⇒ Let $x \gg y$. Let σ be a firing sequence with $(N, M_0)[\sigma\rangle(N, M)$ and $x \in \sigma$. Let $\hat{\sigma}$ be the corresponding firing sequence of σ

and $\hat{M}$ the corresponding marking of M in the augmented system $\hat{S}$. Since $x \in \sigma$, it holds $x^* \in \hat{\sigma}$. Hence, the augmented place p_{x^*}, $a(p_{x^*}) = x$, is marked in $\hat{M}$. As the only transition that consumes a token from p_{x^*}, i. e., transition x, also produces a token in p_{x^*}, this place is marked in all markings $\hat{M}' \in [\hat{N}, \hat{M}\rangle$. Thus, any cut in O that represents the marking $\hat{M}$ or every marking $\hat{M}' \in [\hat{N}, \hat{M}\rangle$ comprises a condition $c_x \in C$ with $a(h(c_x)) = x$. Now consider two cases that follow from $x \gg y$: (1) $y \in \sigma$ or (2) y is not dead in M.

(1) Following on the argument given for transition x, we conclude that place p_{y^*} is marked in $\hat{M}$ and in all markings $\hat{M}' \in [\hat{N}, \hat{M}\rangle$. Then, every cut in O that represents the marking $\hat{M}$ or every marking $\hat{M}' \in [\hat{N}, \hat{M}\rangle$ also comprises a condition $c_y \in C$ with $a(h(c_y)) = y$.

(2) Let $y \notin \sigma$. Then, it holds $y^* \notin \hat{\sigma}$. Since y is not dead in M, y^* is not dead in $\hat{M}$. Hence, there is a firing sequence $\hat{\sigma}_2$ with $(\hat{N}, \hat{M})[\hat{\sigma}_2\rangle(\hat{N}, \hat{M}')$ and $y^* \in \hat{\sigma}_2$. Let t be the last transition of the firing sequence $\hat{\sigma}$ to reach the marking $\hat{M}$. Let $e \in E$ be the event representing the occurrence of t in O, $h(e) = t$. Then, by Theorem 5.3.1, the existence of the firing sequence $\hat{\sigma}_2$ implies that there is an event $f \in E$ with $h(f) = y^*$ and $(e = f) \vee (e \hookrightarrow f)$.

$\Leftarrow$ Let for all events $e \in E$ with $\exists\, c \in Cut(\lceil e \rceil)\ [\ a(h(c)) = x\ ]$ there be either a condition $c \in Cut(\lceil e \rceil)$ with $a(h(c)) = y$ or an event $f \in E$ with $h(f) = y^*$ and $(e = f) \vee (e \hookrightarrow f)$. Let $\hat{M} \in [\hat{N}, \hat{M}_0\rangle$ be a marking in $\hat{S}$ in which place $p_{x^*} \in \hat{P}$ is marked and which is reached by the firing sequence $\hat{\sigma}$. Then, the corresponding firing sequence σ in S contains transition $x \in T$. Let t be the last transition of the firing sequence $\hat{\sigma}$ to reach the marking $\hat{M}$. Let $e \in E$ be the event representing the occurrence of t in O, $h(e) = t$. Then, the cut $Cut(\lceil e \rceil)$ comprises a condition $c_x \in C$ with $a(h(c_x)) = x$. Consider two cases:

(1) There is a condition $c \in Cut(\lceil e \rceil)$ with $a(h(c)) = y$. Then, place p_{y^*} is marked in $\hat{\sigma}$. This place is not marked in $\hat{M}_0$ and y^* is the only transition puts a token into place p_{y^*} without consuming a token from this place. Hence, it holds $y^* \in \hat{\sigma}$ and the corresponding firing sequence σ in S contains transition $y \in T$.

(2) There is an event $f \in E$ with $h(f) = y^*$ and $(e = f) \vee (e \hookrightarrow f)$. By Theorem 5.3.1, the existence of an event continuation implies that there is a firing sequence $\hat{\sigma}_2$ with $(\hat{N}, \hat{M})[\hat{\sigma}_2\rangle$ and $y^* \in \hat{\sigma}_2$. Therefore, y^* is not dead in $\hat{M}$. Consequently, transition y is not dead in the corresponding marking M in S.

□

Algorithm 3: Computation of the co-occurrence relation from the complete prefix unfolding of the augmented system

Input: $S = (N, M_0)$, a bounded system with $N = (P, T, F)$.
$\hat{S} = (\hat{N}, \hat{M}_0)$, the augmented system of S with $\hat{N} = (\hat{P}, \hat{T}, \hat{F})$ and a as the augmentation function.
$\pi = (O, h)$, the complete prefix unfolding of $\hat{S}$ with $O = (C, E, G)$.
Output: $\gg$, the co-occurrence relation of S.

1 Compute order relations $<$, $\#$, and co of O;
2 Compute event continuation relation $\hookrightarrow$ of π (see Algorithm 2);

```
/* Relate events to transitions for which the
   occurrence is indicated by a condition related to
   the augmented place                                  */
```

3 $\mathcal{T}, \mathcal{CA} \longleftarrow \emptyset$;
4 **foreach** $e \in E$ **do** **if** $a(h(e)) \neq \perp$ **then** $\mathcal{T} \longleftarrow a(h(e))$;
5 **foreach** $(e, c) \in (E \times C)$ **do**
6 | **if** $(c \in e\bullet) \vee (e \; co \; c) \wedge ((\bullet c \times \{e\} \subseteq <) \vee (\bullet c = \emptyset))$ **then**
7 | | **if** $a(h(c)) \neq \perp$ **then** $\mathcal{CA} \longleftarrow (e, a(h(c)))$;
8 | **end**
9 **end**

```
// Derive co-occurrence for transitions in S
```

10 $\gg \longleftarrow \emptyset$;
11 **foreach** $(t_1, t_2) \in (\mathcal{T} \times \mathcal{T})$ **do**
12 | check $\longleftarrow$ **true** ;
13 | **foreach** $e \in E$ **do**
14 | | **if** $(e \; \mathcal{CA} \; t_1) \wedge (e \; \mathcal{CA} \; t_2)$ **then**
15 | | | found $\longleftarrow$ **false** ;
16 | | | **foreach** $f \in E$ **do**
17 | | | | **if** $(a(h(f)) = t_2) \wedge ((e = f) \vee (e \hookrightarrow f))$ **then** found $\longleftarrow$ **true** ;
18 | | | **end**
19 | | | **if not** found **then** check $\longleftarrow$ **false** ;
20 | | **end**
21 | **end**
22 | **if** check **then** $\gg \longleftarrow (t_1, t_2)$;
23 **end**
24 **foreach** $(t_1, t_2) \in (T \times T)$ **do**
25 | **if** $t_1 \notin \mathcal{T}$ **then** $\gg \longleftarrow (t_1, t_2)$;
26 **end**

Algorithm 3 shows how the co-occurrence relation is computed for a bounded system given the complete prefix unfolding of the augmented system induced by all of its transitions.

First, we compute the order relations, i. e., the causality, conflict, and concurrency relation, for the complete prefix unfolding (line 1) along with the event continuation relation (line 2).

Second, we extract every transition for which the occurrence of its augmented transition is indicated by a respective event in the complete prefix unfolding (line 4). Here, the idea is to extract all transitions that are not dead in the initial marking of the original net system. Then, we capture dependencies between events and transitions (lines 5 to 9). We check all conditions whether they belong to the cut induced by the event's local configuration. If so, we check whether the condition represents an augmented place of a transition. If this is the case, the relation between the event and the transition to which the augmented place is related is stored in relation $\mathcal{CA}$. The intuition behind is to capture for each event, which transitions have occurred already when the marking represented by the cut induced by the local configuration of the event has been reached.

Third, we exploit the knowledge on this dependency to conclude on co-occurrence for transitions based on Theorem 5.3.6 (lines 10 to 23). That is, we check for all pairs of transitions that are not dead in the initial marking, whether the requirements imposed for co-occurrence by Theorem 5.3.6 are satisfied. Finally, we consider transitions that are dead in the initial marking separately (lines 24 to 26).

Proposition 5.3.7. *Algorithm 3 terminates and after termination* $\gg$ *is the co-occurrence relation of* S.

Proof. Termination: The algorithm iterates over sets that are derived from C, E, and $\mathcal{T}$. The size of set $\mathcal{T}$ is at most the size of E (line 4). Since the net system is bounded, the augmented net system is bounded by Lemma 5.3.5. Hence, the complete prefix unfolding and the sets of events E and conditions C are finite. Thus, the algorithm terminates.
Result: Relation $\mathcal{CA}$ links an event $e \in E$ in the complete prefix unfolding to a transition $t \in T$ in S, if the cut induced by the local configuration of e comprises a condition $c \in C$ that represents the augmented place $p_{t^*} \in \hat{P}$ of t in $\hat{S}$, i. e., $a(h(c)) = t$. Then, we iterate over all pairs of transitions $(t_1, t_2) \in (\mathcal{T} \times \mathcal{T})$ and all events. We rely on relation $\mathcal{CA}$ to check if the cut induced by the local configuration of an event $e \in E$ comprises a condition that represents the augmented place of the first transition t_1. If so, we check whether the requirements of Theorem 5.3.6 are satisfied. Theorem 5.3.6 requires either (1) that the cut induced by the local configuration of e comprises a condition that represents

the augmented place of the second transition t_2 or (2) that there is an event $f \in E$ with $a(h(f)) = y$ and $(e = f) \vee (e \hookrightarrow f)$. Algorithm 3 checks whether both requirements are violated. If no violation is detected, the predicate check remains true and the respective pair of transitions is added to the co-occurrence relation. Finally, transitions in S that are dead in the initial marking of S are treated. For these transitions there is no event in the complete prefix unfolding that represents the augmented transition. Hence, these transitions are characterised as being not in $\mathcal{T}$. □

The algorithm runs in polynomial time with respect to the size of the complete prefix unfolding of the augmented system. Again, we neglect the final step of the algorithm, which sets the co-occurrence relation for dead transitions of the original net system. This is motivated by evaluating the time complexity solely based on the size of the representation of the behaviour, which may be independent from the size of the net system. Therefore, we consider only lines 1 to 23 of Algorithm 3.

Corollary 5.3.8. *The following problem can be solved in* $O(n^4)$ *time with* n *as the number of events and conditions of the complete prefix unfolding:*
For a bounded net system and the complete prefix unfolding of its augmented system, to compute the co-occurrence for all transition pairs of the bounded net system that are not dead in its initial marking.

Proof. Again, we assume all relations used and created in the algorithm to be encoded as bi-dimensional arrays that map to either zero or one. This allows for adding a tuple to the relation or checking its membership in constant time. Computation of the order relations of the complete prefix unfolding is done in $O(|E| \cdot |C|)$ time [243]. By Corollary 5.3.3 the computation of the event continuation relation takes $O(n^4)$ time with n as the number of events and conditions of the complete prefix unfolding. When relating events to transitions, we iterate over $E \times C$, which takes $O(|E| \cdot |C|)$ time. Then, we iterate over $\mathcal{T} \times \mathcal{T}$. Since it holds $|\mathcal{T}| \leqslant |E|$, this takes $O(|E|^2)$ time. As part of an iteration, we may iterate over $E \times E$, which takes $O(|E|^2)$ time in addition. Thus, overall, this part of the algorithm takes $O(n^4)$ time with n as the number of events and conditions of the complete prefix unfolding. □

Again, the polynomial time complexity is relative to the size of the complete prefix unfolding. The latter may be large in size, at most the size of the state space of the net system [151]. Even worse, the augmentation step increases the size of the state space of the net system dramatically. Augmentation by a single transition at most doubles the size of the state space, cf., the reasoning

in the proof of Lemma 5.3.5. Hence, augmentation by all transitions may increase the size of the state space exponentially. This affects the size of the complete prefix unfolding. At a later stage, we will present experimental results to explore this issue.

5.4 IMPLEMENTATION & EXPERIMENTAL RESULTS

To evaluate the presented techniques, we implemented all approaches for the computation of behavioural profiles as part of the *jBPT* library. The library is published under the GNU General Public License (GPL) and available for download[1]. With this implementation, we tested the computation of behavioural profiles for model collections from industry. These experiments provide insights on the structural and behavioural characteristics found for models in an industrial setting. Hence, we are able to judge on the relevance of the assumptions with respect to free-choiceness and soundness imposed by the techniques introduced in Section 5.1 and Section 5.2. The observed absolute computation times also illustrate for which model sizes behavioural profiles are created instantaneously.

In the remainder of this section, we discuss three experiments in detail. For each of the three model collections, we first discuss model characteristics with respect to free-choiceness and soundness. Then, we present results from the application of our techniques for the computation of behavioural profiles. We focus on the approaches for sound free-choice models discussed in Section 5.1 and Section 5.2 for the first two experiments. For the third model collection, we additionally apply the approach for bounded models introduced in Section 5.3. Finally, we shortly summarise the obtained results.

SAP Reference Model

The SAP reference model [95] describes the functionality of the SAP R/3 system. It comprises 604 process diagrams, which are expanded to 737 models in EPC notation as some diagrams contain multiple disconnected EPCs. These EPC models capture different functional aspects of an enterprise, such as sales or accounting.

Model characteristics. A detailed evaluation of the SAP reference model using structural and behavioural measures can be found in [312]. In the following paragraph, we focus on the characteristics that have been relevant for our experimental setup. 23 models are trivial, i. e., they consist of solely one element. Eight out of 737 models show syntax errors that cannot be in-

1 http://code.google.com/p/jbpt/

terpreted unambiguously. Events or functions with more than one incoming or outgoing flow arc are examples for this type of error. Further, the instantiation semantics of EPCs are not formally defined, which raises several questions for models with multiple start events [109]. A class of EPCs with intuitive instantiation semantics has been identified based on the notion of a *start join* [109]. Models with multiple start events contain a start join, if there is *'a join connector such that for every other node* n *in the EPC there is either a path from* n *to the start join or a path from it to* n*.'* [109]. Models without start join may also have intuitive instantiation semantics [361]. Still, investigating the instantiation semantics of these models requires behavioural analysis, whereas the existence of a start join can be decided structurally. Therefore, we require models with multiple start events to have a start join. This requirement is met by 572 process models.

For the models that are non-trivial and free of syntax errors and instantiation issues, we normalised multiple start and end events to create a dedicated entry and exit for each model. For 507 models, such a normalisation was possible without duplicating start or end events. Several models comprise converging OR-connectors, which cannot be mapped to free-choice Petri net constructs, cf., Section 2.3. We replaced block-structured OR-split and OR-join connectors with AND-connectors. This does not affect the behavioural profile, but affects the co-occurrence relation of the causal behavioural profile. Replacing the block-structured OR-split and OR-join connectors with XOR-connectors would have led to the opposite result. In this case, computation would be correct for the co-occurrence relation, but not for the relations of the behavioural profile.

The SAP reference model comprises behavioural errors [472, 316]. From the 507 models that are non-trivial, free of syntax errors and instantiation issues, and have normalised start and end events, 14 models are not sound. We transformed all of the remaining 493 models into sound free-choice WF-systems following on common EPC formalisations [236]. Finally, we assessed whether these systems satisfy the requirements on unstructured net fragments needed to derive the co-occurrence relation from the net structure, see Section 5.1. We encountered two WF-systems that contain a rigid fragment. Both fragments can be mapped to an S-system and, therefore, handled using the presented results.

Computation results. We computed behavioural profiles over all transitions of all 493 WF-systems separately. We grouped the models according to their size, i. e., the number of transitions of the WF-systems. Figure 35 shows the average computation time for each model group in three experiment runs. First, we computed the behavioural profile using the approach introduced in

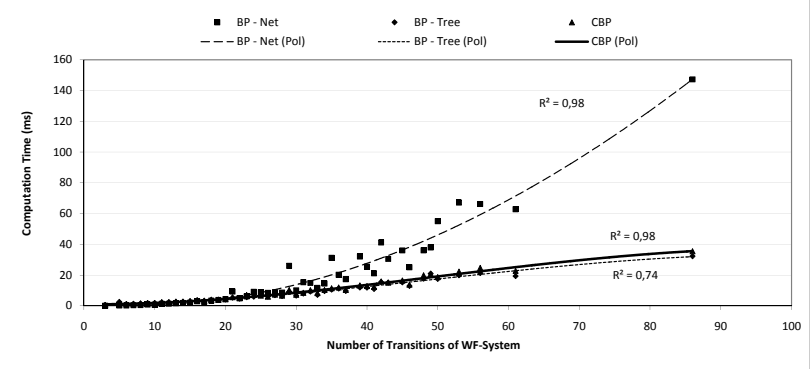


Figure 35: SAP reference model: computation time relative to the size of the WF-system.

Section 5.1 (BP-Net). Second, we derived the same profile using WF-trees as introduced in Section 5.2 (BP-Tree). Third, we computed the causal behavioural profile (including co-occurrence) using WF-trees (CBP). For all three computations, Figure 35 depicts the polynomial least squares regression.

Focussing on the computation of behavioural profiles, the experimental results confirm that the usage of structural decomposition techniques decreases the required computational effort. Computation based on WF-trees (BP-Tree) is faster than leveraging the structure of the net systems directly (BP-Net). The former still relies on the latter to handle rigid net fragments in the system. Further, the computational overhead implied by the co-occurrence relation of the causal behavioural profile is negligible (BP-Tree vs. CBP). For this model collection, computation is done in tens of milliseconds even for the largest models.

Process Models from a Health Insurance Company

The models used for this experiment have been provided by a health insurance company. The models describe the business functions from an organisational perspective and have mainly been applied for staff planning. The collection comprises 1026 process diagrams in EPC notation. Some diagrams contain more than one model, so that the diagrams are expanded to 1042 process models.

Model characteristics. One model shows a syntax error and 16 models are trivial. A large number of models have multiple start and end events. For all except six models, the model structure can be normalised so that each model has a dedicated entry and exit point without duplicating events or functions. All connectors used in the model collection are of XOR- or AND-type. Most models do not show concurrency. Only 48 models comprise

AND-connectors. The models in this collection are virtually free of behavioural errors. Only five models are not sound.

Against this background, we transformed 1014 out of 1042 models into sound free-choice WF-systems [236]. We evaluated the structure of rigid fragments in these systems to see whether the presented approach to the computation of the co-occurrence relation is applicable. There are 142 systems that comprise at least one rigid fragment. All of these fragments are acyclic or can be traced back to S- or T-systems, respectively. Hence, we were able to rely on the approaches for the efficient computation of causal behavioural profiles for all net systems.

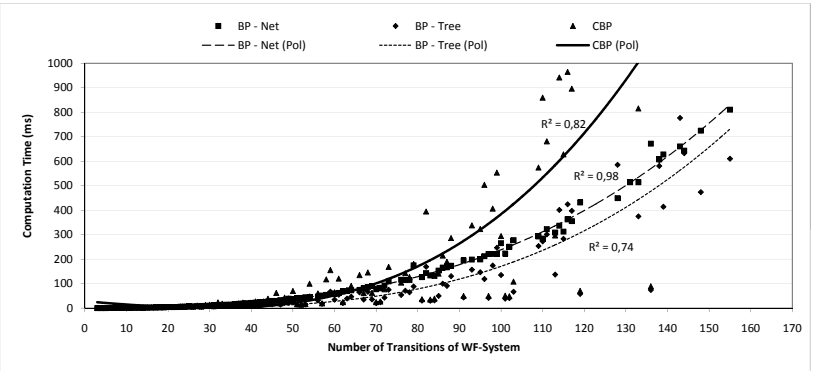


Figure 36: Process models from a health insurance company: computation time relative to the size of the WF-system.

Computation results. As in the previous experiment, we computed the behavioural profiles over all transitions of 1014 WF-systems. We grouped all systems according to their size, i. e., the number of transitions. Figure 36 shows the average computation time for each group of systems up to a size of 170 transitions. Although the largest system contained 456 transitions, only 19 systems had more than 170 transitions and are not covered in Figure 36. We computed behavioural profiles as introduced in Section 5.1 (BP-Net) and Section 5.2 (BP-Tree). Computation of the causal behavioural profile based on WF-trees corresponds to the data series CBP. Figure 36 also depicts the polynomial least squares regression for all three computations.

Again, the experimental results show that the application of structural decomposition techniques speeds up the computation of behavioural profiles. In contrast to the models of the SAP reference model, we observe an overhead for the computation of the co-occurrence relation of the causal behavioural profile. Derivation of causal behavioural profiles takes significantly more time than the computation of behavioural profiles. Nevertheless, the absolute computation times are above one second solely for the 19 large models (with more than 170 transitions) that are not considered in Figure 36.

BIT Process Library

The BIT process library comprises process models that were created in process automation projects in the various industry domains, such as financial services, automotive, telecommunications, construction, supply chain, health care, and customer relationship management. These models were collected and used for a study on soundness verification [156]. For the experiment, we used the parts A, B1, B2, and C of the model collection, a set of 965 models. Originally, all models were captured in the IBM WebSphere Business Modeler[2] in a notation similar to UML activity diagrams. The authors of [156] already provided Petri net formalisations for all models. In fact, all models of the collection were already available as free-choice WF-systems.

Model characteristics. Around one-half of the net systems in this collection are not sound [156]. For the 492 systems that meet the soundness criterion, we investigated the characteristics of rigid net fragments. 207 systems comprise at least one rigid fragment. For 12 out of 207 systems, a least one rigid was cyclic or could not be traced back to S- or T-systems. Hence, our approach to derive the causal behavioural profiles from the net structure is applicable for 476 net systems. Boundedness, the assumption for the computation of behavioural profiles based on complete prefix unfoldings, was met by 924 out of 965 systems.

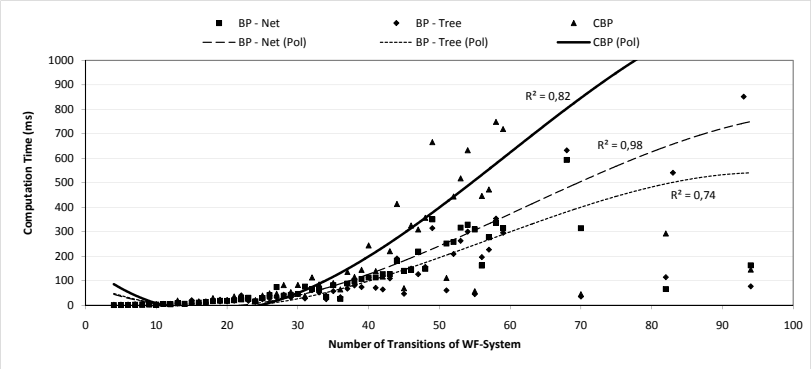


Figure 37: BIT Process Library: computation time relative to the size of the WF-system.

Computation results for sound free-choice WF-systems. We computed the behavioural profiles over all transitions of all 476 WF-systems and grouped the systems according to their size, i. e., the number of transitions of the WF-systems. Figure 37 shows the average computation time for each group of systems up to a size of 100 transitions. Only one system is larger. This system comprises 285 transitions and is not considered in Figure 37. As in the previous experiments, we computed behavioural profiles

2 http://www.ibm.com/software/integration/wbimodeler/entry/

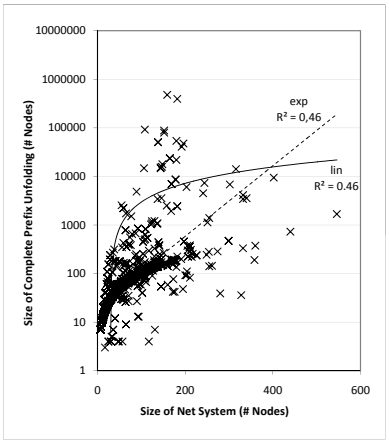
as introduced in Section 5.1 (BP-Net) and Section 5.2 (BP-Tree). The data series CBP refers to the computation of the causal behavioural profile based on WF-trees. For all three computations, Figure 37 depicts the polynomial least squares regression.

The plot supports the observations done for the previous experiments. Computation of behavioural profiles is faster when using the structural decomposition techniques compared to the approach that leverages the net structure directly (BP-Tree vs. BP-Net). Computation of the co-occurrence relation increases the computational effort. Nevertheless, computation is done in hundreds of milliseconds for all models considered in Figure 37. For the largest model with 285 transitions, all computations took around five to 20 seconds.

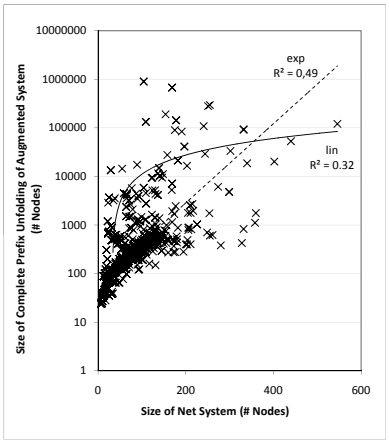
Computation results for bounded WF-systems. Since only one-half of the models in this collection are sound, we also applied the computation of behavioural profiles from a complete prefix unfolding as introduced in Section 5.3. This approach is applicable in a much more general case, since it requires only boundedness of the WF-systems. This requirement is met by 924 out of 965 systems in this collection. The generality of the approach is traded for computational complexity. Construction of the complete prefix unfolding is an NP-complete problem [204, 149] and the complete prefix unfoldings may be large in size, at most the size of the state space of the system [151]. Further, augmentation of WF-systems as used in Section 5.3 to compute the co-occurrence relation from the complete prefix unfolding may increase the size of the state space drastically.

In a first experiment, we focussed on the sizes of the complete prefix unfoldings for the models in this collection. We used the tool *Mole*[3] to generate the prefixes. For three bounded net systems, creation of the prefix was intractable. The maximum size of the derived prefix was around 500.000 nodes. Figure 38 gives an overview of the results. Figure 38a illustrates that prefixes can be an order of magnitude larger in size than the original net systems (the scale is logarithmic). Still, the majority of complete prefix unfoldings was rather small. 94% of the net systems had a prefix with less than 800 nodes. Figure 38a also depicts the linear and exponential least square regressions. Both approximate the relation between the size of the complete prefix unfoldings and the size of the net systems equally well. The implications of the augmentation of WF-systems to compute the co-occurrence relation are illustrated in Figure 38b. For 19 net systems, computation of the complete prefix unfolding was not possible after augmentation. The maximum size of a prefix of an augmented system was around 900.000 nodes. Figure 38b shows that the augmentation step leads to prefixes that are larger in size.

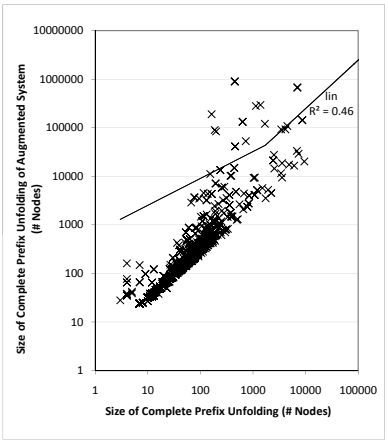
3 http://www.lsv.ens-cachan.fr/~schwoon/tools/mole/



(a) Size of the complete prefix unfolding relative to the size of the net system.



(b) Size of the complete prefix unfolding of the augmented system relative to the size of the net system.



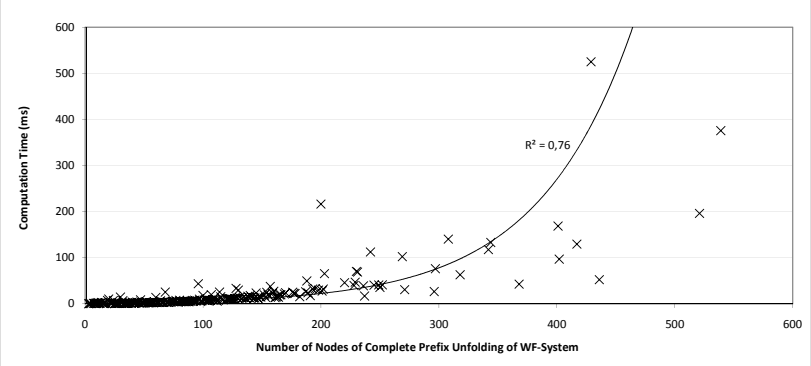
(c) Size of the complete prefix unfolding of the augmented system relative to the size of the complete prefix unfolding.

Figure 38: Size of the complete prefix unfoldings derived for the net systems of the BIT Process Library.

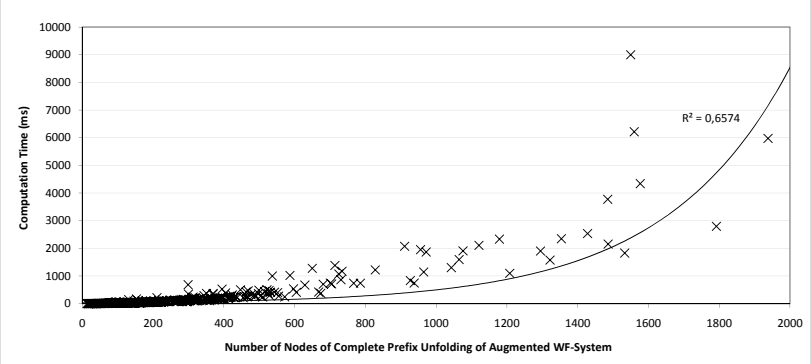
Figure 38c directly relates the size of the prefix of the original net systems to the size of the prefix of the augmented systems. Apart from notable outliers, the relation is linear for most systems, though.

In a second experiment, we focussed on the actual computation of behavioural profiles from the complete prefix unfolding. We used the implementation of the algorithms presented in Section 5.3 that have been published as part of the *jBPT* library. Those rely on the *Unfolding-based Model Analyzer (UMA)*[4] to generate the complete prefix unfolding. For performance reasons,

4 http://www.service-technology.org/uma



(a) Time for the computation of the behavioural profile relative to the size of the complete prefix unfolding of the original net system.



(b) Time for the computation of the causal behavioural profile relative to the size of the complete prefix unfolding of the augmented net system.

Figure 39: Computation times for (causal) behavioural profiles.

UMA was configured to abort unfolding of branching processes once an unsafe place was encountered in the original net system. Hence, the computed behavioural profiles are solely approximated for net systems that are bounded but unsafe. Apart from this restriction, for 11 net systems the computation of the causal behavioural profile turned out to be intractable. For the remaining bounded net systems, Figure 39a and Figure 39b illustrate the time needed to compute behavioural profiles (Figure 39a) or causal behavioural profiles (Figure 39b) based on the prefix of the original net system or its augmented counterpart. In either case the computation times includes the time needed to create the prefix. Both plots also show the exponential least squares regression. Behavioural profiles could be computed in less than a second for all considered net systems. This low computation time is explained by the rather small size of the prefixes (less than 600 nodes). The large prefixes observed in Figure 38 are not completely unfolded by *UMA*. Computation of the causal behavioural profile takes significantly longer, since the augmentation step increases the size of the prefixes. Seven prefixes had more than 2000 nodes, the largest having around 5500 nodes, and are

not visualised in Figure 39b. The increase in prefix size leads to computation times of up to several seconds. For the largest prefix with around 5500 nodes, the computation took slightly less than 30 seconds.

Summary of the Experimental Results

The motivation for the presented experiments was twofold. On the one hand, we wanted to judge on the relevance of the assumptions with respect to free-choiceness and soundness that are required to compute behavioural profiles efficiently. On the other hand, we wanted to investigate absolute computation times.

Free-choiceness turned out to be non-critical. Only the SAP reference model contained elements that require non-free-choice constructs in the Petri net formalisation. However, those elements were only block-structured OR connectors, so that a replacement with AND-connectors (XOR-connectors) for the computation of the behavioural profile (causal behavioural profile) can be used as a workaround. Besides a few models in the SAP reference model, the soundness criterion was violated in a large scale solely in the BIT process library. Here, our efficient algorithms could be applied to only one-half of the models. Further, the normalisation of models to arrive at a single entry and exit for a model (i. e., the WF-net structure) turned out to be a problem. Several models in the SAP reference model could not be treated due to their complex instantiation semantics (i. e., the models without start join). As acknowledged by [109, 361], however, the question of how to interpret these models is a separate research area. Looking at all three model collections, we conclude that our assumptions on free-choiceness and soundness are met by the majority of models. Also, we observed only a fee unstructured net fragments that are cyclic and cannot be traced back to S-systems or T-systems. We conclude that the assumptions of the approach to compute causal behavioural profiles using WF-trees are of minor importance in practise.

Our experiments showed that behavioural profiles are computed within milliseconds for WF-systems that are sound and free-choice. Hence, for this class of models derivation of behavioural profiles is done instantaneously. Our experiments with the algorithms for bounded net systems revealed that computation is done in less than two seconds if the complete prefix unfolding has less than around 1000 nodes. Taking into account that 94% of the net systems of the BIT process library have a prefix with less than 800 nodes, this approach works for the majority of models. Nevertheless, we also faced several models for which computation took several seconds or was even intractable.

5.5 RELATED WORK

The techniques for the computation of behavioural profiles introduced in this chapter relate to techniques for the derivation of other behavioural relations which we discussed in Section 4.5.

The behavioural relations introduced for the verification of hardware specifications [393] are derived by parsing an acyclic program into a program tree. This is similar to our approach of leveraging the RPST decomposition technique to compute behavioural profiles. Still, the causal behavioural profile comprises behavioural details that go beyond the relations introduced in [393]. In addition, we also presented techniques for the computation of behavioural profiles that are able to cope with cyclic net systems. The relations used for service match making [144] are also directly derived from a parse tree of a BPEL process. Again, our technique based on the RPST decomposition technique is close to this approach, whereas our other techniques generalise the computation for net systems that are not structured in terms of single-entry and single-exit subnets.

Behavioural relations that are used in the context of process mining are typically derived from observed sequences of transition occurrences, not from a net system. Apparently, occurrence sequences may be generated from a net system by means of *playout* [197, 446]. Assuming a certain level of completeness of the generated behaviour, it allows for the derivation of behavioural relations. Finding an appropriate level of completeness is known as the rediscoverability problem in process mining [457, 446]. The causal matrix used in genetic process mining [459, 106] is derived from the net structure. Given a transition, its directly preceding and directly succeeding transitions are captured in the causal matrix. For the follows and precedes relations used to judge on the quality of mined process models [395], no efficient computation algorithm is available to the best of our knowledge.

We already discussed that the order matrix [271, 272, 273, 274] virtually coincides with the behavioural profile. The order matrix is computed by directly traversing a block-structured process model and extracting the respective relations in quadratic time to the size of the model [274]. Following the ADEPT modelling paradigm [97], such a block-structured process model is sound by construction. Therefore, this class of process models corresponds to sound free-choice net systems that do not show unstructured net fragments. Our computation using structural decomposition techniques goes beyond the results of [274], as it allows for selective computation of the behavioural relations. We are able to determine the relation for a single pair of transitions in linear time to the size of the model.

Computation of causal footprints leverages a set of structural rules [470, 473]. As there is no unique causal footprint of a model, it depends on the granularity of the intended behavioural abstraction which techniques are selected for computation. Starting with local rules that exploit the neighbourhood of nodes that split or merge the control flow, computation of the closure of causal footprints may be applied to obtain a richer causal footprint [470]. This approach requires soundness of the model and is computationally hard in the general case. The techniques introduced in this chapter, in turn, provide a classification of requirements on the net systems to achieve efficient computation of the behavioural abstraction. In the class of sound free-choice net systems all computations, except for co-occurrence in unstructured cyclic subnets, are done in polynomial time to the size of the net system.

Further related work comprises the application of the concepts leveraged for the computation of behavioural profiles in other contexts. The RPST has been introduced with the focus on a mapping between process languages [481]. Tree-based decompositions have also been used to refactor process models [361, 480] and for control-flow analysis [223, 479, 360], process comparison [256], pattern application in process modelling [187], and process model abstraction [357].

Since the unfolding technique has been introduced by McMillan [305], it has been extended and investigated in a large number of publications, see [149] for a thorough discussion. The unfolding technique has been applied for various purposes. Unfoldings are used to check properties of net systems such as reachability of certain markings, or for LTL model checking [148]. Also, domain specific problems, e. g., the analysis and synthesis of asynchronous circuits [232] or restructuring of process models [361], have been addressed using the unfolding technique.

5.6 CONCLUSION

We dedicated this chapter to the computation of behavioural profiles. Behavioural profiles are computed efficiently for sound free-choice WF-systems. The relations of the behavioural profile are determined in low polynomial time to the size of the net system. The co-occurrence relation is computed efficiently for T-systems, S-systems, and acyclic sound free-choice WF-systems. We complemented these results by leveraging structural decomposition techniques for the computation of behavioural profiles. The combination of these techniques yields an algorithm that computes the relation of the behavioural profile for a transition pair of a sound free-choice WF-system in linear time to the size of the net. If one restriction is satisfied – unstructured net frag-

ments are T-systems, S-systems, or acyclic – we are able to compute even the causal behavioural profile for a transition pair in linear time to the size of the net.

For net systems that do not satisfy these requirements, we presented an alternative computation technique. Under the assumption of boundedness of the net system, we derive behavioural profiles from the complete prefix unfolding. These prefixes are a compact representation of the system's state space. The general applicability of this approach is bought for computational complexity. Creation of the complete prefix unfolding is computationally hard. Further, the computation of the co-occurrence relation relies on an augmentation of the net system, which may increase the size of the state space exponentially. Finally, we tested the implementation of all techniques with three model collections from industry. The results suggest that the assumptions for the efficient computation techniques are met by the majority of process models observed in practise. Using these techniques, behavioural profiles are computed within milliseconds. We also illustrated the applicability of the approach based on the complete prefix unfolding of a net system. Although we faced systems for which computation was intractable, the majority of the tested net systems could be handled within less than two seconds.

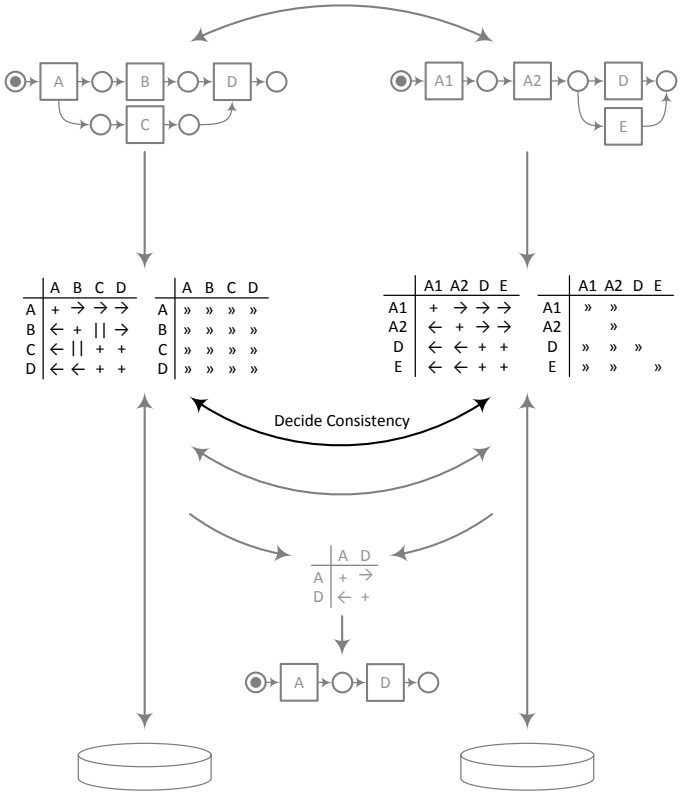
We conclude that a behavioural profile indeed provides an abstraction of the behaviour of a net system that is computed efficiently.

Part III

CONSISTENCY ANALYSIS

6 DECIDING PROCESS MODEL CONSISTENCY

This chapter is based on results published in [505, 493].



This chapter introduces the first part of our framework for the analysis of behaviour consistency. We focus on deciding process model consistency using Boolean criteria based on behavioural profiles. First, we motivate the application of behavioural profiles in Section 6.1. To this end, we review different options for a formal grounding of consistency analysis, i. e., behaviour equivalences and behavioural relations. Then, Section 6.2 proposes several Boolean consistency criteria. Those leverage behavioural profiles and are interrelated. They define a spectrum of consistency criteria. Section 6.3 turns the focus on the application of these consistency notions in a concrete setting. For the evaluation of consistency between business centred process models and technical workflow models, we report on findings from an experiment on the consistency perception of process modelling experts. We review related work in Section 6.4 and conclude this chapter in Section 6.5.

6.1 CONSISTENCY NOTIONS

This section first introduces a set of example net systems. Those depict the same business operations and have been aligned by correspondences between their transitions. For these process models, behaviour consistency has to be assessed, e. g., for validating whether the support provided by information systems meets the business level concerns, see Section 1.3. As stated before, consistency may be interpreted as the *absence of contradictions* [533]. According to the same author, the question of how to assess the *absence of contradictions* is a verification problem. After we introduced the example net systems, we review elementary properties that may be verified to conclude on behaviour consistency. First, we focus on behaviour equivalences and their application for consistency evaluation. Second, we elaborate on how behavioural relations are used for consistency analysis. Third, we narrow the scope to the application of behavioural profiles to decide behaviour consistency.

An Example Setting

We illustrate the different options to decide behaviour consistency with a lead-to-order process. Figure 40 depicts three net systems, all capturing this process, along with correspondences between (sets of) their transitions. The net systems show a similar processing of a lead. Once the contact details have been obtained, the potential customer is contacted and a quote submission is prepared. Then, the quote is submitted, which may be followed by a negotiation phase. Assuming that the net systems have been created for different purposes, however, they also show several differences. For instance, system (a) captures the process only until the submission of a quote. Subsequent steps such as the negotiation of a contract are neglected. Systems (b) and (c) capture the processing until a deal is settled. Still, there are differences between both models. The reception of a request for quote is captured in system (c), but neglected in system (b).

Consistency Notions based on Behaviour Equivalences

To decide whether two process models show the same behaviour, notions of behaviour equivalence may be applied. As discussed in Section 4.4, these notions are classified in the linear time – branching time spectrum [474, 476], see also [363, 209]. Behaviour equivalences compare process models that feature exactly the same set of activities. Following the terminology introduced in Section 3.1, this requires a correspondence relation between

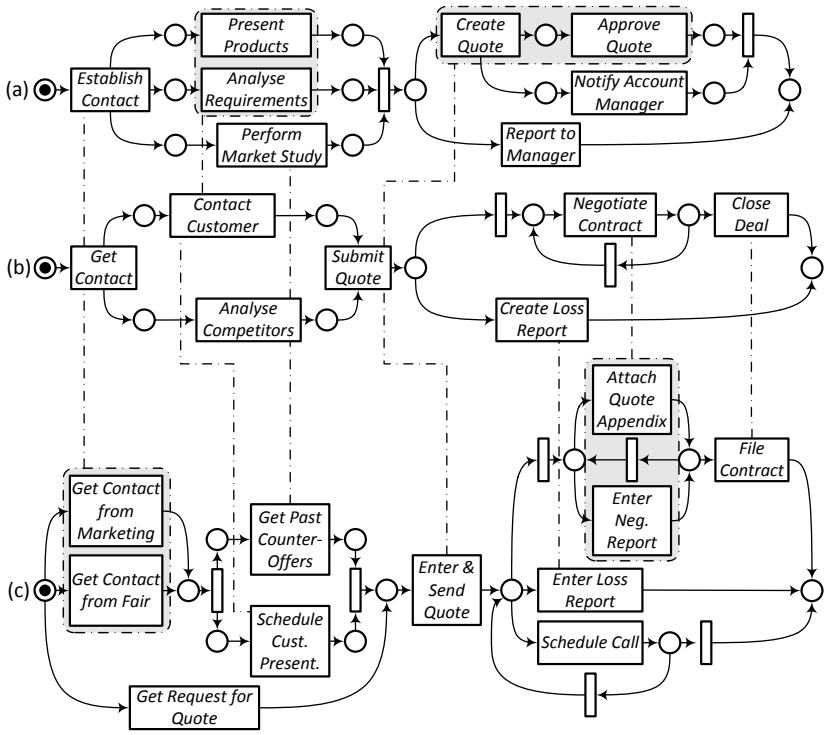


Figure 40: Three net systems depicting a lead-to-order process that are aligned by correspondences.

two process models to be bijective. The examples introduced earlier illustrate that this requirement cannot be assumed to hold in the general case. Two questions have to be answered, before a notion of behaviour equivalence is applied to judge on process model consistency: How to cope with activities that are not aligned and how to cope with complex correspondences?

To approach these questions, we refer to the concepts of *refinement* and *extension*, as they have been introduced for specialisations of behavioural models [410, 411, 412]. Refinement refers to the definition of an activity (or a set thereof) in more detail. Extension refers to the act of adding new activities. Hence, activities that are not part of any correspondence are an extension. A complex 1:n correspondence can, to a certain extent, be interpreted as a refinement. This interpretation assumes that the correspondence implies semantic equivalence of the related sets of activities. Further, refinements are directed and hierarchical, i. e., they cannot explain overlapping correspondences. Despite these differences, the notions of refinement and extension provide an angle to review literature on the application of behaviour equivalences in our setting.

Extension. Common equivalences are not invariant under extensions, which can be seen as the dual operation of a *forgetful refinement* [477]. Removing an activity from a process model may break behaviour equivalence. The question of how to still

assess behaviour equivalence in the presence of extensions has mainly been addressed under the term *behaviour inheritance*. Behaviour inheritance aims at transferring the concept of inheritance known for static structures, e. g., for class diagrams in UML, to the level of behavioural models. In particular, behaviour inheritance has been investigated for object life cycles that describe the behaviour of software artefacts.

Whether an extension of an object life cycle is behaviour preserving is assessed either on the *observed* behaviour or the *invocable* behaviour [138]. Following this line, Basten and van der Aalst proposed two elementary notions of behaviour inheritance, *protocol inheritance* and *projection inheritance*, and combinations thereof for net systems [451, 33].

Informally, two systems satisfy projection inheritance, if they are branching bisimilar once transitions that are not part of any correspondence are considered to be silent. In other words, transitions in one system that are without counterpart in the other system are *hidden*. Branching bisimulation is insensitive to silent transitions and, thus, closely related to stuttering equivalences known from model checking, see [69, 30].

Two systems satisfy protocol inheritance, informally speaking, if they are branching bisimilar once transitions that are not part of any correspondence are removed from the system. In contrast to projection inheritance, protocol inheritance requires all transitions that are not part of any correspondence to be *blocked*.

Similar ideas have been presented by Schrefl and Stumptner for Object Behavior Diagrams (OBDs) [410, 411, 412], a behavioural model that comprises activities and explicit states. They introduce *observation consistency*, which corresponds to projection inheritance. Note that, in contrast to trace equivalence, observation consistency refers not only to transitions (OBD activities) but also to states. Further, there exists the notion of *invocation consistency* for OBDs, which is close to protocol inheritance.

Refinement. There is a large body of work on equivalence preserving refinements for Petri nets, refer to [64] for a thorough survey. Such a refinement is always assumed to be hierarchical. A place or transition of a net system is replaced by a subnet, so that the subnet is *embedded* into the original net [351, 329]. Hence, a refinement leads to equally directed non-overlapping 1:n correspondences. One may argue that complex correspondences between process models that can be traced back to one of the known refinement operations should be considered to be consistent. Still, the operationalisation of such an approach is challenging, if the set of transitions related to a correspondence is not forming an isolated subnet. Therefore, behaviour preserving reduction techniques [329] may be required to judge whether those transitions can be traced back to a refinement op-

erator. We are not aware of any work that traces back arbitrary correspondences between behavioural models to refinements.

One may think of a different approach to cope with complex correspondences following on the idea of stuttering equivalences [69]. These equivalences acknowledge that a single state transition may have been refined into a sequence of state transitions. In our context, the occurrence of a set of transitions, all belonging to one complex correspondence, could be traced back to the occurrence of a single transition. In prior work [501], we followed a similar approach. To lift the notions of behaviour inheritance to the level of complex correspondences, we used the partitioning of traces that is induced by complex correspondences. This approach does not impose any restrictions on the cardinality or direction of correspondences. Still, it assumes non-overlapping correspondences and considers only complete traces, i. e., traces from the initial to the final state of the process.

We revisit the example net systems introduced in Figure 40. Here, transitions that are not aligned by any correspondence may be blocked or hidden as proposed by notions of behaviour inheritance. Focussing on systems (a) and (b), we see that both options often need to be combined. The unlabelled transition merging three parallel branches in system (a) has to be projected, as blocking it would stall the process. In contrast, transition 'Report to Manager' in system (a) represents an alternative option to continue processing, which is not captured in system (b). This suggests blocking the transition for assessing behaviour consistency. Further, the net systems show complex correspondences. Some even involve sets of transitions that do not form an isolated subnet, e. g., transitions 'Create Quote' and 'Approve Quote' in system (a). Hence, to decide whether these correspondences can be traced back to behaviour preserving refinements is not straight-forward. Apart from that, we may rely on the approach that lifted behaviour inheritance to the setting of complex correspondences based on a partitioning of traces [501]. Assuming a suitable combination of hiding and blocking of transitions that are not aligned, both alignments, between systems (a) and (b), and between (b) and (c), are consistent.

We summarise that behaviour consistency may be decided based on behaviour equivalences. Still, the presence of complex correspondences imposes various challenges towards to the application of behaviour equivalences, which have been addressed only partially in the literature – overlapping correspondences are not covered. Further, our examples illustrate that an appropriate combination of hiding and blocking of transitions that are not aligned may be required to conclude on consistency.

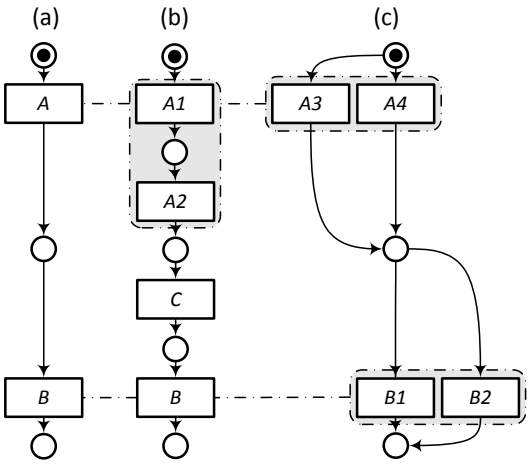


Figure 41: Aligned net systems to illustrate the choice of behavioural relations for consistency analysis.

Consistency Notions based on Behavioural Relations

Notions of behaviour consistency can be based on behavioural relations instead of behaviour equivalences. Taking a certain definition of relational semantics, consistency of aligned process models is decided as follows. All relations for tuples of transitions in one model are compared to the relations for tuples of corresponding transitions in the other model. Although n-ary behavioural relations may be considered in principle, most existing behavioural relations are binary; they are defined for pairs of transitions. Hence, consistency assessment translates into comparing relations for all pairs of transitions that are part of correspondences.

In Section 4.5, we reviewed several relational semantics. The major difference between them is their focus on either direct causal dependencies, e. g., the footprint comprising the relations of the α-algorithm [457, 446], or indirect dependencies, such as the behavioural profile. We explicated this difference with the look-ahead that is assumed during computation of the relations. We may exploit direct successorship, a look-ahead of one, or indirect dependencies, a far-look-ahead.

Against the background of alignments between related process models, indirect dependencies seem to be more suited to assess consistency. Alignments can be expected to be partial. Activities that are not relevant regarding the modelling purpose will not be captured in a process model, i. e., they are extensions when comparing two related process models. However, consistency assessment based on relations that focus on direct successorship is sensitive to extensions. Indirect dependencies, in turn, are not affected by extensions. Consider the alignment between systems (a) and (b) depicted in Figure 41. The trans-

itions A and B in system (a) correspond to the transitions $\{A1, A2\}$ and B in system (b). Relations that built upon direct successorship capture an order dependency for the pair (A, B) in system (a), $A \rightarrow B$ according to the footprint [446]. There is no order dependency for any of the pairs of corresponding transitions in system (b), $A1 \# B$ and $A2 \# B$ according to the footprint. Systems (b) and (c) in Figure 41 also illustrate that complex correspondences have to be considered separately. The dependency $A1 \# B$ observed in system (b) is not mirrored by any of the pairs of corresponding transitions in system (c).

These issues are avoided by using indirect dependencies. Taking the relations of the behavioural profile, the strict order dependency between transitions A and B in system (a), is also observed for all pairs of corresponding transitions in system (b), $A1 \rightsquigarrow B$ and $A2 \rightsquigarrow B$. The same holds for the alignment between system (b) and (c). Hence, consistency assessment based on behavioural profiles is insensitive to extensions of process models and can be applied for complex correspondences in a straightforward manner.

Behavioural profiles are a behavioural abstraction – the captured dependencies neglect causal dependencies between transition occurrences. This may be problematic for certain scenarios of deciding behaviour consistency. The usage of causal behavioural profiles countervails this effect. Still, these profiles provide an abstraction and only approximate trace semantics. For illustration, consider again Figure 41. Co-occurrence between transitions A and B in system (a) is also observed between pairs of corresponding transitions in system (b). In system (c), however, we do not observe co-occurrence between the respective pairs of transitions. In this system, co-occurrence is not manifested in a binary relation.

Consistency Notions based on Behavioural Profiles

The application of a behavioural abstraction, such as behavioural profiles, implies a certain information loss when deciding consistency. Even if causal behavioural profiles are used, trace semantics of the respective models is only approximated. However, there is evidence that even more coarse-grained abstractions are suited for consistency analysis.

Certain drivers of process modelling, such as process understanding and communication, tend to yield *happy path* process models that only capture the most frequent execution sequence of a process [291, 21, 320]. Such models abstract from alternative branches that are of minor importance for understanding the overall processing. For the setting in Figure 40, for instance, such an alternative branch would be reception of a request for

quote in system (c), which is without counterpart in systems (a) and (b). Also, system (c) incorporates the possibility that a customer is not answering to a submitted quote. System (b) abstracts from this deviation from the standard processing. Capturing such alternative branches potentially breaks causal dependencies between pairs of activities. 'Contact Customer' is a cause of 'Submit Quote' in system (b), whereas there is no such dependency between the corresponding transitions in system (c). The usage of behavioural profiles for consistency analysis allows for neglecting such dependencies explicitly. Extensions, such as the reception of a request for quote in system (c), do not affect the indirect order dependencies as defined by the behavioural profile. This suggests exploiting only the relations of the behavioural profile, and to neglect co-occurrences, when deciding behaviour consistency in a certain context.

The phenomenon of *happy path* process models also suggests that repetitions are only modelled on a certain level of detail. They may be abstracted in high-level business process models that give an overview of business operations. *Coupled repetitive activities* are a structural pattern occurring frequently in the refinement of process models [239]. Accordingly, information on potential repetition may be neglected for consistency analysis. This aspect is easily incorporated into the consistency assessment based on behavioural profiles. All self-relations are excluded from the consistency analysis.

Coming back to the example in Figure 40, both alignments respect the behavioural profiles. The dependencies observed in one system, including self-relations, are also present for all pairs of corresponding transitions in the other system. With respect to co-occurrence, we observe several deviations. Besides the aforementioned violations caused by extensions, however, co-occurrence violations are also caused by complex correspondences. For instance, the co-occurrence dependency between 'Get Contact' and 'Contact Customer' in system (b) is not present for the corresponding transitions in system (c).

To conclude, we discussed that consistency assessment may be based on behaviour equivalences or behavioural relations. The former imposes various challenges with respect to complex correspondences. Approaches based on behaviour equivalences can be expected to be computationally hard. In particular, the choice to either block or hide transitions that are not aligned adds to the computational complexity. As an alternative, behaviour consistency may be decided using behavioural relations. To this end, the presence of extensions and complex correspondences suggests utilising indirect dependencies. Further, we motivated that even a relatively coarse-grained behavioural abstraction may be suited for consistency analysis.

6.2 BEHAVIOURAL PROFILE CONSISTENCY

The previous section sketched how behavioural profiles are used to decide consistency of an alignment. This section is dedicated to the formal definition of consistency notions. We show how behavioural profile equivalence introduced in Section 4.4 is applied for net systems that are aligned by correspondences. Then, we discuss a spectrum of consistency criteria based on behavioural profiles. Finally, we focus on the interpretation of confidence values assigned to correspondences, see Section 3.1, when applying the proposed consistency criteria.

Definitions of Consistency Criteria

The definition of consistency notions uses some auxiliary concepts. We rely on the notion of a correspondence relation as it has been introduced in Definition 3.1.1. Essentially, a correspondence relation between two transitions associates pairs of corresponding transitions of two net systems to each other. As such, it defines elementary correspondences and, implicitly, also complex correspondences, see Section 3.1.

First, we need the notion of aligned transitions. Those are transitions of an aligned net system that are part of a correspondence.

Definition 6.2.1 (Aligned Transitions)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, and $\sim \subseteq T_1 \times T_2$ a correspondence relation. The set of *aligned transitions* $T_1^{\sim} \subseteq T_1$ of S_1 is defined as $T_1^{\sim} = \{t_1 \in T_1 \mid \exists\ t_2 \in T_2\ [\ t_1 \sim t_2\ ]\}$. The set $T_2^{\sim}$ of S_2 is defined analogously.

Further, we need a notion of equivalence for the relations of behavioural profiles of different net system. This equivalence relates to the type of the profile relation and is defined for all relations of the behavioural profile along with the reverse strict order relation.

Definition 6.2.2 (Type Equivalence of Profile Relations)
Let $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2\}$ be behavioural profiles. Two relations $R_1 \in \mathcal{B}_1 \cup \{\rightsquigarrow_1^{-1}\}$ and $R_2 \in \mathcal{B}_2 \cup \{\rightsquigarrow_2^{-1}\}$ are *type equivalent*, denoted by $R_1 \simeq R_2$, iff either

- $R_1 = \rightsquigarrow_1\ \wedge R_2 = \rightsquigarrow_2$,
- $R_1 = \rightsquigarrow_1^{-1}\ \wedge R_2 = \rightsquigarrow_2^{-1}$,
- $R_1 = +_1\ \wedge\ R_2 = +_2$, or
- $R_1 = \|_1\ \wedge\ R_2 = \|_2$.

Using these concepts, we define consistency notions based on behavioural profiles. Given two aligned net systems, behavioural

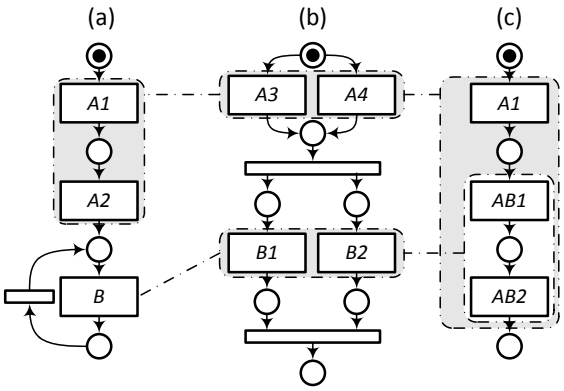


Figure 42: Example net systems: all alignments are weak behavioural profile consistent; only the alignment between (b) and (c) is behavioural profile consistent; none of the alignments is causal behavioural profile consistent.

profile consistency requires that the profile relations observed for two transitions of different correspondences in one model are type equivalent for all pairs of corresponding transitions in the other model. As motivated in the previous section, there may be reasons to relax this preservation of behaviour with respect to repeated execution of activities. Therefore, we first introduce weak behavioural profile consistency, which neglects the self-relations of aligned transitions.

Definition 6.2.3 (Weak Behavioural Profile Consistency)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, and $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2\}$ their behavioural profiles. Let $R_1 \in \mathcal{B}_1 \cup \{\rightsquigarrow_1^{-1}\}$ and $R_2 \in \mathcal{B}_2 \cup \{\rightsquigarrow_2^{-1}\}$. A correspondence relation $\sim \subseteq T_1 \times T_2$ is *weak behavioural profile consistent*, iff for all transition pairs $(t_x, t_y) \in (T_1^{\sim} \times T_1^{\sim})$, $t_x \neq t_y$, and transitions $t_s, t_t \in T_2^{\sim}$, $t_s \neq t_t$, $t_x \sim t_s$, $t_y \sim t_t$, it holds that either (1) $(t_x R_1 t_y \wedge t_s R_2 t_t) \Rightarrow R_1 \simeq R_2$ or (2) $t_x \sim t_t$ and $t_y \sim t_s$.

We illustrate weak behavioural profile consistency with the example net systems depicted in Figure 42. Consider the systems (a) and (b). There are two complex correspondences, one between the sets of transitions $\{A1, A2\}$ and $\{A3, A4\}$, and one between transitions $\{B\}$ and $\{B1, B2\}$. In the presence of complex correspondences, weak behavioural profile consistency requires the preservation of behavioural relations for all transition pairs that are induced by the complex correspondence. Every combination of transitions that are part of different correspondences is checked. For this example, we observe strict order between transitions $\{A1, A2\}$ and transition B in system (a). This is consistent with system (b), as it holds $A3 \rightsquigarrow B1$, $A4 \rightsquigarrow B1$, $A3 \rightsquigarrow B2$, and $A4 \rightsquigarrow B2$. The relation between transitions belonging to the same correspondence is not considered. Although we ob-

serve $A1 \rightsquigarrow A2$ in system (a) and $A3 + A4$ in system (b), this difference does not affect the consistency criterion according to Definition 6.2.3. Further, the difference in the behavioural profile with respect to self-relations, $B \| B$ in system (a), but $B1 + B1$ and $B2 + B2$ in system (b), is neglected. The alignment between systems (a) and (b) is weak behavioural profile consistent.

Weak behavioural profile consistency is also applicable in the presence of overlapping correspondences. For illustration, consider the systems (b) and (c) in Figure 42. Here, there are two complex correspondences, one is defined between transitions $\{A3, A4\}$ and $\{A, AB1, AB2\}$, and one between transitions $\{B1, B2\}$ and $\{AB1, AB2\}$. Both overlap on transitions $AB1$ and $AB2$ in system (c). Still, the alignment between systems (b) and (c) is weak behavioural profile consistent. The strict order dependencies between transitions $\{A3, A4\}$ and $\{B1, B2\}$ in system (b) are also observed in system (c) for the respective transitions. Relations between transitions that are part of the overlap are not considered according to Definition 6.2.3. For instance, the relation between $t_x = A4$ and $t_y = B2$ in system (b) is not required to hold for $t_s = AB2$ and $t_t = AB1$ in system (c) since it holds $A4 \sim AB1$ and $B2 \sim AB2$.

We strengthen the consistency criterion to take potential repetition of activities into account. Behavioural profile consistency extends weak behavioural profile consistency by considering also self-relations of transitions.

Definition 6.2.4 (Behavioural Profile Consistency)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, and $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2\}$ their behavioural profiles. Let $R_1 \in \mathcal{B}_1 \cup \{\rightsquigarrow_1^{-1}\}$ and $R_2 \in \mathcal{B}_2 \cup \{\rightsquigarrow_2^{-1}\}$. A correspondence relation $\sim \subseteq T_1 \times T_2$ is *behavioural profile consistent*, iff it is weak behavioural profile consistent and for all transitions $t_x \in T_1^{\sim}$ and $t_s \in T_2^{\sim}$, $t_x \sim t_s$, it holds $(t_x R_1 t_x \wedge t_s R_2 t_s) \Rightarrow R_1 \simeq R_2$.

Behavioural profile consistency requires the self-relations for all transitions in one system to be type equivalent to the self-relations of the corresponding transitions in the other system. For the examples depicted in Figure 42, this yields the following results. The alignment between systems (a) and (b) is no longer considered to be consistent. The potential repetition of transition B in system (a) is not reflected in system (b), as transitions $B1$ and $B2$ may occur at most once. The potential repetition of one of these transitions would not suffice to meet behavioural profile consistency. The self-relations have to be type equivalent for all corresponding transitions. The alignment between systems (b) and (c) is behavioural profile consistent as all aligned transitions of both systems may occur at most once.

In the same vein, the co-occurrence relation of the causal behavioural profile may be required to hold for all corresponding transitions of an alignment.

Definition 6.2.5 (Causal Behavioural Profile Consistency)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, and $\mathcal{CB}_1 = \{\rightsquigarrow_1, +_1, \|_1, \gg_1\}$ and $\mathcal{CB}_2 = \{\rightsquigarrow_2, +_2, \|_2, \gg_2\}$ their causal behavioural profiles. A correspondence relation $\sim \subseteq T_1 \times T_2$ is *causal behavioural profile consistent*, iff it is behavioural profile consistent and for all transition pairs $(t_x, t_y) \in (T_1^{\sim} \times T_1^{\sim})$, $t_x \neq t_y$, and transitions $t_s, t_t \in T_2^{\sim}$, $t_s \neq t_t$, $t_x \sim t_s$, $t_y \sim t_t$, it holds that either (1) $t_x \gg_1 t_y \Leftrightarrow t_s \gg_2 t_t$ or (2) $t_x \sim t_t$ and $t_y \sim t_s$.

Causal behavioural profile consistency extends behavioural profile consistency by considering the co-occurrence relation for pairs of aligned transitions. As the identity relation over transitions of a net system is subsumed by the co-occurrence relation, see Section 4.2, we refer only to pairs of distinct transitions.

Causal behavioural profile consistency implies a rather strict interpretation of how to check the co-occurrence relation. For sets of transitions that are part of a correspondence it induces an *all-or-none* semantics. Given two correspondences $c_1 = (T_1', T_2')$ and $c_2 = (T_1'', T_2'')$, co-occurrence between two transitions $t_1' \in T_1'$ and $t_2' \in T_2'$, $t_1' \gg t_2'$, requires that all pairs of corresponding transitions $t_1'' \in T_1''$ and $t_2'' \in T_2''$ are co-occurring. This, in turn, implicitly requires that also all transitions $T_1' \times T_2'$ are co-occurring. Hence, either all or none of the pairs of transitions of two sets of aligned transitions of a net system have to be co-occurring. If this property is not met, causal behavioural profile consistency cannot hold for an alignment that comprises complex correspondences between two net systems.

One may think of a weaker interpretation of how to check co-occurrences. Co-occurrence between two transitions of two correspondences in one system may be required to hold only for *one* pair of corresponding transitions in the other system. Since the causal behavioural profile captures only co-occurrences between pairs of transitions, however, this does not seem to be appropriate. Taking systems (a) and (b) of Figure 42, the co-occurrence from transition B to transitions A1 and A2 in system (a), would be violated in system (b) as it holds B1 $\not\gg$ A3, B1 $\not\gg$ A4. This may be considered to be an anomaly as one of the corresponding transitions A3 or A4 has to occur before transition B1. However, addressing this anomaly would require defining co-occurrences over sets of transitions, e. g., the occurrence of B1 implies the occurrence of one of the transitions $\{A3, A4\}$. This is not feasible as n-ary co-occurrence relations may be observed. Hence, we acknowledge this limitation and stick to the strict interpretation of how to consider co-occurrence given in Definition 6.2.5.

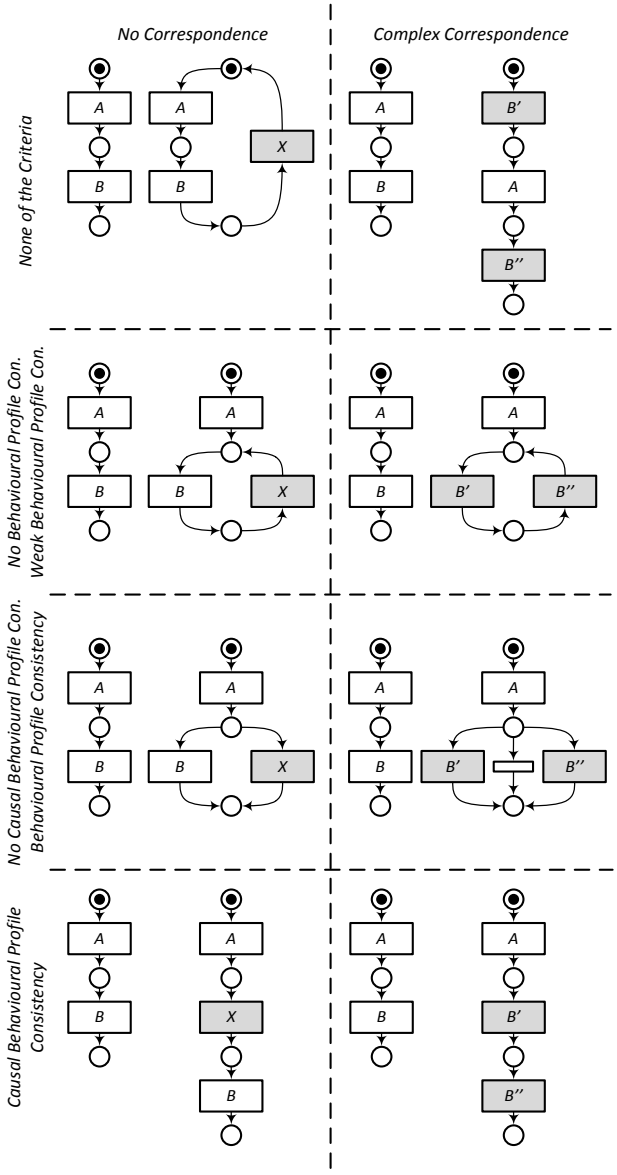


Figure 43: Overview of the spectrum of consistency criteria. We assume correspondences between the systems to be defined according to the transition labels.

A Spectrum of Consistency Criteria

The consistency criteria introduced earlier step-wise increase the amount of information, i. e., the number of transition pairs and relations, on which the consistency analysis is based. As a consequence, these criteria define a consistency spectrum.

We illustrate this spectrum in Figure 43. In each cell of the table, this figure shows two small net systems for which we assume correspondences to be defined according to the transition labels. The first column shows examples in which one transition is not part of any correspondence, transition X. The second column contains examples that comprise a complex correspondence, between transition B and transitions $\{B', B''\}$. The align-

ments in the first row violate all presented criteria. In both cases, the strict order dependencies between transitions A and B is not in line with the model that shows an additional transition X or two transitions $\{B', B''\}$ related to B.

The next column of Figure 43 shows examples for which the correspondences satisfy the weakest consistency criterion, weak behavioural profile consistency. For all models in this column, we observe a strict order dependency between transition A and transition B (or B' and B''). However, there are differences with regard to the potential repetition of transition B (or B' and B'').

The third column shows examples for which the alignment is behavioural profile consistent. All relations of the behavioural profile of the first system coincide with the profile relations of the second system for corresponding transitions. In both cases, there are still differences related to co-occurrence.

The last column illustrates systems for which the alignment satisfies also the strictest consistency criterion, causal behavioural profile consistency.

Confidence of Deciding Consistency

The presented consistency criteria exploit behavioural relations for all pairs of corresponding transitions that are defined by the correspondence relation of an alignment.

In Chapter 3, we discussed that the construction of an alignment between process models typically leverages textual, structural, or behavioural similarities of model elements or parts of the process models, respectively. This is considered in our notion of an alignment, which comprises a correspondence relation and a confidence function, see Section 3.1. The latter captures the quality of an elementary correspondence. This value typically directly follows from the similarity assessment that was used to construct the alignment.

The introduced consistency criteria are Boolean notions. The confidence associated to the decision taken by one of these consistency criteria is the average of the confidence values for all elementary correspondences. In other words, the confidence in the consistency decision directly follows from the confidence that the correspondences between two net systems hold.

6.3 CONSISTENCY PERCEPTION

The previous section introduced a spectrum of formal notions of behaviour consistency. These notions provide means to adjust consistency requirements towards a concrete setting. In this section, we narrow our scope and focus on consistency between process models that capture a business centred view on business

operations and those that have been created for process automation. We already discussed these two drivers for process modelling in Section 1.2. Models created for these two purposes are likely to show a variety of differences. Those stem, among others, from the constraints an IT infrastructure imposes on *how* the goal of a business process is achieved, whereas business centred modelling typically focusses on *what* needs to be done. The missing fit between business centred process models and technical workflow models is often referred to as the 'Business-IT Gap' [70, 186, 388]. Further, the focus on this kind of process models results in a clear motivation for deciding consistency, see the drivers of consistency analysis in Section 1.3. A business centred process model is typically used as a specification against which a workflow model is validated.

Within this scope, this section investigates the question of *which formal notion of behaviour consistency can best approximate perceived consistency of modelling experts*. In relation to formal notions, we leverage the notions of the consistency spectrum outlined before. There has been anecdotal evidence for the proximity and remoteness of these various notions to the human consistency perception. Eventually, however, the question of proximity can only be validated from an empirical perspective involving experts in process modelling. Addressing this demand, we report the findings from an online experiment. This experiment can be classified as a correlational study similar to [32, 154, 317]. We identified 69 expert statements from process analysts from all over the world, and we analysed how the aforementioned consistency criteria match the perceived consistency of our subjects.

In the remainder of this section, we first derive a set of hypotheses on the proximity of the consistency criteria to the consistency perception of modelling experts. Then, we present our research design and introduce the results. Finally, we discuss the results, reflect on potential threats to validity, and draw implications.

Hypotheses on Consistency Notions

Our hypotheses refer to the outlined spectrum of consistency criteria based on behavioural profiles. Here, causal behavioural profile consistency can be seen as the baseline. It is the strictest criterion that most closely approximates trace semantics of the models aligned by correspondences. In Section 6.1, we motivated the usage of weaker consistency criteria with the phenomenon of *happy path* process models. Therefore, we investigate the following consistency criteria, *causal behavioural profile*

consistency, *behavioural profile consistency*, and *weak behavioural profile consistency*.

We formalise our arguments in three hypotheses on the proximity between the criteria and the perception of experts on the consistency of a pair of process models.

*H*1: Pairs of process models that are *causal behavioural profile consistent* will be perceived to be more consistent than pairs that are not.

*H*2: Pairs of process models that are *behavioural profile consistent* will be perceived to be more consistent than pairs that are not.

*H*3: Pairs of process models that are *weak behavioural profile consistent* will be perceived to be more consistent than pairs that are not.

To assess these hypotheses in an experimental setup with modelling experts, additional factors have to be taken into account. It is well known that personal differences influence model comprehension [77]. In particular, modelling experience [352, 372] and study background [315] have been found to be relevant comprehension factors. It is important to check how these additional factors affect the results. Therefore, we aim to investigate personal characteristics of the experts including work status in academia or industry, work focus on business or IT, or time of modelling experience. We formulate the following questions to address the potential influence of personal differences.

- Does perceived consistency of academic and industry experts differ?
- Does perceived consistency of business and IT experts differ?
- Does perceived consistency of experts with longer modelling experience differ from those with shorter modelling experience?

Answers to these questions allow us to conclude on the extent to which our findings are independent of additional factors.

Research Setup

To test the set of introduced hypotheses, we conducted an experiment on the perception of consistency notions using an online questionnaire. In the following paragraphs, we introduce our research setup. We elaborate on the target audience, the selection and creation of pairs of process models used in the questionnaire along with their characteristics concerning behaviour consistency, and the experiment instrumentation.

Subjects / Target Audience. We aimed at testing the hypotheses in the most general setting. Our target audience were practitioners involved in the creation, analysis, or implementation of

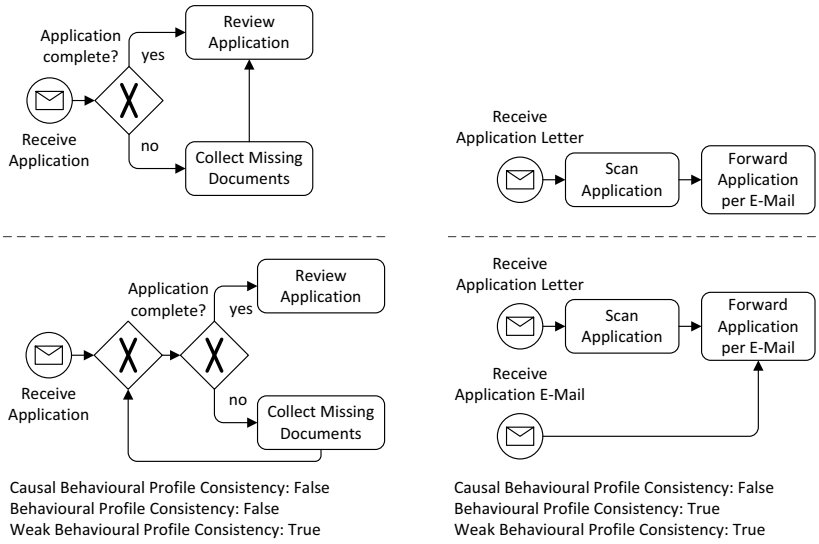


Figure 44: Two objects of our experiment illustrating a difference related to an activity (event, respectively) that is (a) part of a correspondence, (b) not part of a correspondence.

business process models that have significant real-world experience in process modelling. Thus, we decided to gather data on their consistency perception using an online questionnaire as timely access to a wide range of participants with different backgrounds was crucial for the external validity of our investigations. It was not feasible to approximate the entire audience for the questionnaire. Hence, we could not select a representative sample and had to rely on convenient sampling and a self-recruiting questionnaire design, following on the recommendations given in [87].

Objects / Pairs of Process Models. To assess the perception of the subjects on process model consistency, the questionnaire comprised 11 pairs of process models, the objects of our experiment. Each pair consisted of process models in BPMN [2], which contained at most four activities and used only a small subset of BPMN modelling constructs, such as plain and message start events, activities, and XOR and AND gateways. These constructs are widely used and can be seen as the common core of process modelling languages [538]. For two models including a boundary error event and a multiple-instance activity, we included annotations explaining the respective semantics. The objects have been taken from two generally known domains to not have any domain bias, namely processing of a purchase quote and processing of a job application. However, activity labels have been chosen rather abstract like 'Create Quote' or 'Receive Applica-

tion'. That, in turn, should ensure that participants are not influenced by any process context.

In all 11 objects, the respective process models showed a slight difference as illustrated in Figure 44. The selection of differences has been guided by classifications of process model differences available in related work, see [206, 123]. In prior work, we reviewed these classifications and identified nine common differences between process models that (1) relate to the activity or control-flow perspective and (2) can be expected to be observed between business centred process models and technical workflow models [498]. Examples for these differences are *activity fragmentation* or *decision distribution*. Each difference may affect behaviour consistency, so that each was covered by at least one object in our experiment. Two differences were included twice. Discussions with process modelling experts in a pre-test suggested considering these differences in two variants. Out of the 11 objects, six showed a difference related to activities that are part of a correspondence. For the remaining five pairs, the difference was related to activities that are not aligned by a correspondence. Thus, we ensured that both types of differences are considered equally. All objects used in the experiment can be found in [493].

Consistency Factors. For each object we determined, whether the consistency criteria referred to in Hypotheses *H1* to *H3* hold. This yields three dichotomous factors: Causal Behavioural Profile (CBP) Consistency, Behavioural Profile (BP) Consistency, and Weak Behavioural Profile (WBP) Consistency. We saw that five, seven, and nine pairs of process models were CBP consistent, BP consistent, and WBP consistent, respectively. Hence, all consistency criteria are affected by the selection of process model differences based on existing classifications. Concerning the distribution of the different criteria, our selection of models reflects the fact that most process model pairs can be assumed to meet at least the weakest criterion. On the other hand, our selection is balanced for the strictest criterion, causal behavioural profile consistency, which is satisfied by nearly one-half of the model pairs.

Instrumentation. We conducted a pre-test that was succeeded by a round of discussion with 8 practitioners. Their feedback led to the adaptation of the pairs of process models, an update of the introductory description, and revised annotations on execution semantics. We also used the feedback to choose the items to be asked for each process model pair. We considered the option that experts might judge the consistency of the process models based on both, closeness and equivalence. Therefore, it was found most appropriate to ask 'I think the processes are similar' (strong disagreement to strong agreement) and 'I think the processes represent the same real-world process' (strong dis-

agreement to strong agreement). By having these two separate items, we aimed to avoid a bias towards a particular position underlying the experts' perceptions of consistency.

Once these changes were implemented accordingly, we published the questionnaire online and left it available for three months. An introductory text discussed the need of having different process models for different purposes of a common process. No further information on notions of behaviour consistency was given to the participants to focus on the pure consistency perception. The Web site of the questionnaire was advertised via channels having an affinity towards process modelling. These included the SAP Developer Network, the German BPM-Netzwerk, special interest groups on the social business platform XING, BPMN-related blogs such as BPMN.info, academic mailing lists as isworld and the German WI list, and finally via process consulting and vendor companies.

Demographics & Results

In the following, we introduce the results obtained with the experiment. We describe demographics, distribution of perceptions, hypotheses testing, and checks for potential interactions.

Demographics. The online questionnaire was filled out by 157 persons. We performed several steps to guarantee that the analysis data will be of high quality. We considered only answers of participants who completed the full questionnaire. Then, we excluded those entries that where obviously filled out in a rush (answer time less than 10 minutes) and those where participants got distracted by other tasks (answer time more than 100 minutes). Our pre-test with 8 practitioners revealed that it is not possible to reasonably complete the questionnaire in less than 10 minutes. Participants with more than 100 minutes obviously took a break, which might distort their performance data. Further, we excluded those participants that classified themselves as students since we aimed to get expert opinions. Finally, there were also cases of persons giving the same consistency assessment to all model pairs. Again, this pattern points to a thoughtless clicking through the online questionnaire, such that we excluded these answers.

The remaining data included the answers of 69 participants. Those originated from 27 countries from all over the world with a focus on the United States (15) and Germany (14). Each participant assessed the 11 pairs of process models such that we have 759 data points altogether. 20% of the participants used the *German version* of the questionnaire and 80% the *English version*. The *answer time* varied between 11 and 85 minutes with a mean of 23 and a standard deviation of 15. Only four participants took

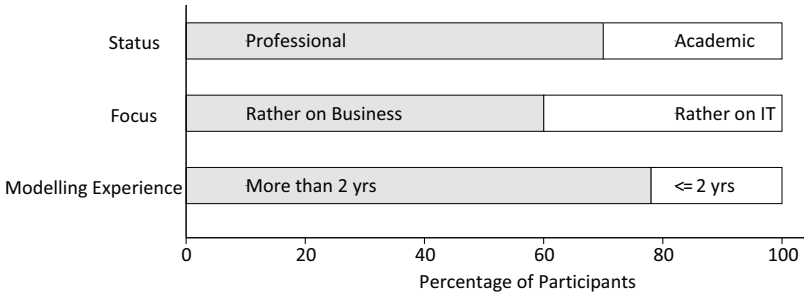


Figure 45: Demographics for the participants of our experiment (expert statements).

more than 50 minutes. As illustrated in Figure 45, 30% of the participants classified themselves as *academics* and 70% as *practitioners* (answers by students were excluded). 60% stated they would rather work on the *business side* of process management and 40% rather on the *IT side*. Only 22% had less than two years, whereas 78% had at least two years of *modelling experience*. These numbers provide us with confidence that we actually collected the assessments of experts.

Distribution of Perceptions. In the questionnaire, we asked for two assessments for each model pair on a 1 to 5 Likert scale ('I think the processes are similar' and 'I think the processes represent the same real-world process') to measure the consistency perception. We observed that participants provided consistent answers to these questions. Cronbach's Alpha as a reliability check yielded a good value of 0.76 such that we summed up the two factors into one scale variable called *Perceived Consistency* ranging from 2 to 10. Its mean value was rather central with 6.44 and with a standard deviation of 2.195. We performed the Kolmogorov-Smirnov test, which indicated that its values are normally distributed. This is an important criterion for making analysis of variance (ANOVA) testing applicable.

Hypotheses Testing. To investigate Hypotheses *H1* to *H3*, we first derived descriptive statistics and constructed boxplots for the *Perceived Consistency* and each of the three consistency notions. Then, we inspected rank correlations and examined differences in variance of *Perceived Consistency* for each of the notions.

The boxplot for *Perceived Consistency* and Causal Behavioural Profile Consistency indicates partial support for the theoretical assumption behind Hypothesis *H1*, see Figure 46a. When considering the median, model pairs that meet the criterion are perceived to be more consistent by the participants. However, the range covered by the lower and upper quartiles is equal and the mean value is virtually equal for the models that meet the criterion and those that do not, see Table 1. The boxplot and the descriptive statistics for *Perceived Consistency* and Behavioural Profile Consistency are in line with Hypothesis *H2*, see

Table 1: Descriptive statistics for *Perceived Consistency*.

	Mean	Median	Standard Deviation
All Objects	6.44	6.00	2.195
Causal Behavioural Profile Consistency	6.47	7.00	2.136
No Causal Behavioural Profile Consistency	6.41	6.00	2.245
Behavioural Profile Consistency	6.55	7.00	2.170
No Behavioural Profile Consistency	6.23	6.00	2.228
Weak Behavioural Profile Consistency	6.62	7.00	2.167
No Weak Behavioural Profile Consistency	5.63	6.00	2.148

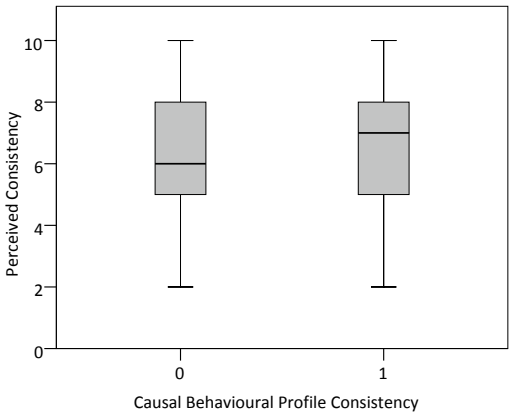
Table 2: Support for Hypotheses *H1* to *H3*.

	H1	*H2*	*H3*
F-Statistic	0.135	3.816	23.469
Significance	.713	.051	.000
Support	no support	no support	strong support

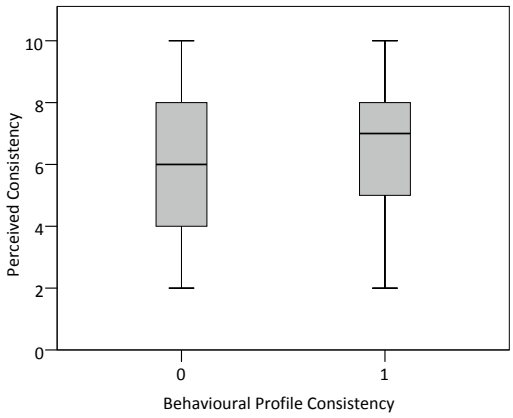
Figure 46b. The model pairs with consistent profile relations are also perceived to be more consistent. A similar observation is made for Weak Behavioural Profile Consistency, see Figure 46c and Table 1. In contrast to the first boxplot in Figure 46a, the plots for the criteria based on the non-causal behavioural profile indicate a bigger variance in perception if the model pair is inconsistent according to these criteria. In particular, the lower quartile of the perceived consistency is larger meaning that more participants tended to give low consistency values. This difference in the consistency perception is also reflected in the mean values given in Table 1.

As a next step, we inspected whether the observed differences are significant from a statistical point of view. We identified a minimal positive Spearman rank correlation [429] between Causal Behavioural Profile Consistency and *Perceived Consistency* of 0.007 which is insignificant (p=.855). There is a positive correlation between Behavioural Profile Consistency and *Perceived Consistency* of 0.068 (p=.059), and also a significant positive correlation between Weak Behavioural Profile Consistency and *Perceived Consistency* of 0.171 (p=.001) at a significance level of 99%. These facts are not in line with *H1* and only partially with *H2*. They comply with the assumptions of *H3*.

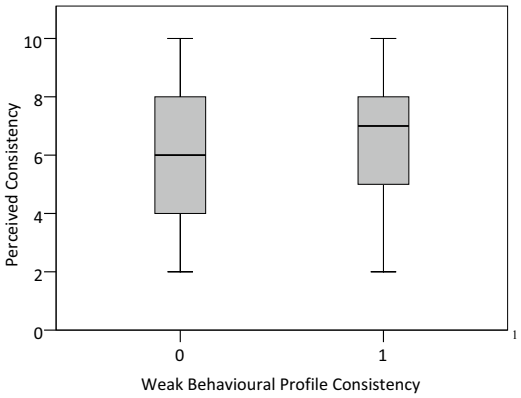
Additionally, we performed analysis of variance tests for each of the consistency notions and *Perceived Consistency*, see Table 2. None of the Levene statistics was significant, such that homogen-



(a)



(b)



(c)

Figure 46: Perceived Consistency vs. (a) Causal Behavioural Profile Consistency, (b) Behavioural Profile Consistency, and (c) Weak Behavioural Profile Consistency.

eity of variance can be assumed. For Causal Behavioural Profile Consistency and *Perceived Consistency* the difference is not significant with p=.713 and an F of 0.135. This fact points to the rejection of *H1*. For Behavioural Profile Consistency and *Perceived Consistency* the significance of the difference is below the 95%

Table 3: ANOVA model for *Perceived Consistency*.

			CBP Consistency	BP Consistency	Weak BP Consistency
	F-Stat.		1.997	0.003	6.804
	Sign.		.158	.957	.009
Status	F-Stat.	0.001	1.062	0.612	0.100
	Sign.	.976	.303	.434	.752
Focus	F-Stat.	2.891	2.641	2.112	0.605
	Sign.	.035	.048	.097	.612
Modelling	F-Stat.	0.504	0.321	0.050	1.313
Experience	Sign.	.680	.811	.985	.269

level with p=.051 and an F of 3.816. This does not support *H2*. For Weak Behavioural Profile Consistency and *Perceived Consistency* the difference is significant with p=.000 and an F of 23.469. This fact supports *H3* strongly.

Potential Interactions. Finally, we considered the set of moderating factors and checked potential interactions between consistency notions and the personal variables *status* (academic or practitioner), *focus* (business or IT), and *modelling experience* (less than two years or at least two years). For this purpose, we calculated an ANOVA for *Perceived Consistency* using these three personal variables and the three notions as input. Table 3 shows the most important variables of this ANOVA model and their interactions. It can be seen that Weak Behavioural Profile Consistency is the most significant factor with more than 99% significance. Focus is also significant as a factor. That stems from the fact that IT persons gave slightly higher consistency perceptions than business participants. There is no significant interaction effect of any of these variables on the relation between each of the three consistency notions and *Perceived Consistency* except for Causal Behavioural Profile Consistency and focus. We inspected the data in further detail and found that business people showed a slightly positive correlation between consistency perceptions and Causal Behavioural Profile Consistency, while there was no connection for IT people (neither positive nor negative). Both correlations were insignificant.

Even though only one of the background factors is significant and there is only one significant interaction, our results show that consistency perception is not entirely independent of the personal background. As mentioned before, business people tend to judge the consistency of our experiment objects on a lower level. Against this background, it is remarkable that Hypothesis *H3* is still supported with more than 99% significance.

The correlation between *Perceived Consistency* and Weak Behavioural Profile Consistency is very strong, such that differences with respect to *Perceived Consistency* between business and IT people are of no consequence. The significant interaction related to Causal Behavioural Profile Consistency and work focus can be expected to stem from the same observation. There are fewer objects that show Causal Behavioural Profile Consistency than objects that show the other two consistency criteria. Hence, the fact that business people gave lower consistency values yields a correlation with Causal Behavioural Profile Consistency, which is still not significant.

Discussion of Results

Our results on testing Hypothesis *H1* suggest that causal behavioural profile consistency is not suited to judge on the consistency of business centred process models and technical workflow models. Although this criterion is based on a behavioural abstraction, it turns out to be too strict in this context. Rejection of Hypothesis *H1* suggests that causal behavioural profile consistency and the consistency perception are not correlated. Hence, consistency analysis of business centred process models and technical workflow models based on this criterion may lead to results that are not intuitive for process stakeholders.

In our experiment, aspects that are abstracted by the causal behavioural profile with regard to trace semantics have not been visible in the used objects. Consequently, all objects that were causal behavioural profile consistent were also *projection compatible* [501]. The latter criterion is based on trace equivalence, hides all activities that are not aligned following the idea of projection inheritance [451, 33], and copes with complex correspondences based on trace partitioning. As such, it lifts the trace equivalence criterion directly to alignments of process models. However, trace equivalence was introduced mainly for program verification and investigations on the design of programming languages [213]. These use cases impose different requirements than the consistency analysis of business centred process models and technical workflow models. The equivalence of programs requires all execution paths to be available in both equivalent behaviour specifications. However, we already discussed that process models designed for communication tend to focus at least partially on the *happy path* [291, 21, 320]. Accordingly, as modelling experts are aware that not all execution options are explicitly captured, the perception of consistency seems to closer correlate with the weaker notions of the consistency spectrum. Still, an appropriate combination of blocking and hiding of activities as discussed for the examples in Section 6.1 may be an alternative

way to cope with this issue. Blocking activities also excludes certain execution paths when deciding behaviour equivalence.

The tests for the other two hypotheses illustrate that the criteria that leverage only the relations of the non-causal behavioural profile are more suited in our context. They better approximate the consistency perception of process modelling experts. Even though the tests for *H2* are not significant at the 95% level, weak behavioural profile consistency (*H3*) led to a highly significant correlation with the perceived consistency. It is important to see that consistency requirements are relaxed by these criteria in a well-motivated manner. The *freedom of contradictions* between the behaviour of two process models is interpreted in a way that the order of potential activity occurrences has to be respected, whereas causal dependencies are considered to be negligible. Any conclusion in the sense 'the more relaxed a notion of consistency, the closer to expert judgement', therefore, must not be drawn.

Further, important findings of our experiment relate to the personal background of the participants of the questionnaire. The status (academia or industry) does not have an influence on the consistency perception. The argument that people from academia would be stricter in their consistency assessment potentially due to a formal background and less experience with real-world settings is not supported by our data. In addition, modelling experience did not turn out to have an effect for our results. Our results are twofold with respect to the fact, whether a participant classifies their work to reside on the business side or the IT side of process management. Although this difference is often called to account for failing process management efforts, its influence on the consistency perception of process models illustrating these different perspectives is reflected solely in the absolute level of consistency values. Hence, there seems to be a common understanding with respect to the inevitability of certain differences between business centred process models and technical workflow models. Nevertheless, business people tend to give lower consistency values for our objects in general. We consider this to be a remarkable, yet unexpected, observation. IT-people are typically concerned with concrete implementations of business processes, whereas business people stay on the rather conceptual side. Hence, the latter could be expected to show a more relaxed understanding of consistency.

Potential Threats to Validity

There are some threats to validity of our study. We want to highlight self-recruitment and the selection of experiment objects.

Earlier, we mentioned that we did not try to apply a random sampling, but relied on self-recruitment. This implies a certain threat to representativeness. Our strategy towards good external validity was to make sure that the sample of participants that we considered in our statistical evaluation can actually be regarded as modelling experts with some confidence. Our assumption in this context is that modelling experts in general share a common understanding of what consistency is. If there were different schools of thought that led to diverging perceptions on consistency, there might be a bias in our data if these schools were not covered in a representative way. However, we do not possess any evidence that such differences exist. In our various discussions with modelling experts, we observed a strong agreement among them on consistency matters.

The selection of model pairs could introduce a potential threat to validity. To investigate the consistency perception of process modelling experts, we focussed on a concrete setting, i. e., behaviour consistency between business centred process models and technical workflow models. Hence, we assume that the differences presented in [206, 123, 498] are representative for the deviations between models created for these two purposes. Two of these classifications explicitly aim at capturing differences in this context [206, 498]. We are not aware of an exhaustive validation of their relevance and completeness, though. Further, it may be the case that perceptions of consistency vary with the size and complexity of process models. The pairs that we choose were rather compact in this dimension. The choice was made for different reasons. First, a pair of process models should show a single difference as classified in related work [206, 123, 498]. In this way, we ensured that any obligations of a participant towards a dedicated difference would have a minor influence on the result. In such a case, the perceived consistency would be affected for one object instead of for multiple objects. Second, we would have run into complex interactions if we had varied the size and complexity of the pairs. It is well known from research on process model metrics that size and complexity have a negative effect on process model comprehension [312]. This implies that larger models would likely introduce a substantial level of noise in the data, as the chance of bad comprehension and, therefore, inadequate consistency assessment rises with complexity. Consequently, we selected rather small model pairs to limit the cognitive effort of the participants and reduce the risk of participants leaving the questionnaire without completion.

Implications

We want to highlight major implications of our study. Those relate to the application of behavioural abstractions for consistency analysis, the importance of utility considerations for formal criteria in Business Process Management (BPM), and the development of process modelling tools.

We discussed in Section 6.1 that consistency analysis may also be based on behaviour equivalences. However, our study revealed that, in a certain context, rather weak consistency criteria show a good approximation of experts' consistency perceptions. The extent to which this observation depends on the purpose for which the process models have been created and on the driver of consistency analysis has to be evaluated in further studies. It will be interesting to see whether our observation can be confirmed in a different context.

Further, our research highlights the importance of empirical research on formal correctness criteria for process management. This observation is not limited to scenarios where consistency matters. Many formal correctness criteria such as *soundness* [448] or *relaxed soundness* [116] have been defined with a clear motivation of utility. This utility can be evaluated to a great share using empirical research methods. Up until now, virtually no experiments have been conducted to prove their utility. We foresee that a feedback loop between research on formal properties and the evaluation of their usefulness in a certain context may be of great benefit to advance the field of BPM. Our results provide a concrete starting point for further empirical research in this area. The observation of business people evaluating the consistency of process model pairs on a lower level than people with an IT background clearly needs further investigation.

Finally, current process modelling tools rarely support the definition of an alignment between models representing different abstractions of a common process. If so, there are typically no means of consistency analysis. Our findings can be seen as a stimulus for the realisation of consistency analysis in commercial process modelling tools based on behavioural relations.

6.4 RELATED WORK

Our work on deciding consistency for aligned process models relates to two streams of research, view-oriented approaches to modelling of processes and systems, and interaction consistency.

Model Views

We motivated the analysis of behaviour consistency with the presence of multiple process models that capture (overlapping parts of) the same business process. These models provide different views on business operations, created for a certain purpose. These views evolve independent of each other, which requires constructing an alignment and deciding behaviour consistency a posteriori.

View-oriented process modelling takes a different approach to accommodate for the multitude of potential perspectives on a business process. To this end, the existence of a holistic core model is assumed. Then, the model is adapted towards a dedicated purpose and group of stakeholders by deriving customised *process views* [280, 145, 536, 55]. In the same vein, methodologies for integrated system design propose to derive technical realisations from business models directly by means of refinements [206, 21, 241]. Although these approaches do not start with a core model, the final technical model can be seen as a core model as well. Then, the business models are interpreted as process views on the final technical model.

Views created by these approaches are consistent by construction. Consistency between a base model and a process view is ensured, as the latter is derived by means of rules. Often these rules concern the model structure. Sill, such structural consistency is motivated by guaranteeing a certain degree of behaviour consistency in terms of *'activity orderings'* [280]. Albeit often not mentioned explicitly, the latter is typically decided using notions of behaviour equivalence. As a consequence, notions of behaviour inheritance that we reviewed in Section 6.1 are often implicitly respected. This also holds for techniques that integrate multiple views of a behavioural model, such as [366, 343, 314].

Similar observations can be made for Software Engineering techniques that target the development of complex systems using multiple *viewpoints*, e. g., [167, 56, 61, 334, 163, 126]. Viewpoints realise a separation of concerns by focussing on different aspects of the system to be built. For viewpoints, the authors of [56, 61] advocate the application of behaviour equivalences and partial preservation of traces as notions of behaviour consistency. Behaviour equivalences are also used to assess consistency in [126]. Other approaches favour the description of behavioural dependencies imposed by each viewpoint in terms of logic statements. Then, the conjunction of these statements is checked for satisfiability [334, 163].

View-oriented modelling avoids the pitfalls addressed in this thesis by restricting the evolution of related models. Once process views or viewpoints are derived via a limited set of trans-

formation rules, there is no need to construct an alignment and decide behaviour consistency. However, these rules typically induce only a certain type correspondences between model elements – hierarchical refinements. Then, correspondences are non-overlapping and equally directed. Further, there are no n:m complex correspondences and for 1:n correspondences the associated set of activities is structurally restricted. The activities are often required to form a single-entry single-exit region. In the light of the variety of process modelling drivers, it does not seem to be realistic to assume that such a restricted creation and evolution of process models is feasible in all cases. It is interesting to see that even in system design the inevitability of inconsistencies between different views on the system has been acknowledged in the literature [163, 335, 238]. According to [238], *'non-hierarchical transformations are not rare exceptions. In the transition from high-level models of the application domain to the implementation model, we find them anywhere'*.

Interaction Consistency

Behaviour consistency is also a central concept once the scope of a single organisation is left and inter-organisational processes are modelled, e. g., in the domain of business-to-business (B2B) integration. Such a setting imposes consistency requirements at different stages of the inter-organisational integration [409]. Public processes of interacting partners have to be consistent, either [301]. Further, consistency must be ensured between these public processes and their private implementations [300, 465]. Consistency of interactions has been studied extensively [532, 43, 300, 301, 110, 79, 535, 465, 282, 112].

There are fundamental differences between our notions of behaviour consistency and those used for interacting processes. For interacting processes, the interaction protocol is in the centre of interested when deciding the *freedom of contradictions*. Ensuring that two protocols are not contradicting may require considering the moment of choice [478]. This suggests using rather strict notions of behaviour equivalence. On the other hand, the assumed communication model, see [230], may allow for certain behavioural deviations that are not in line with common behaviour equivalences [110]. As an example, consider a protocol that defines a concurrent asynchronous sending of two messages. A second protocol that specifies a sequential reception of these two messages will typically be considered to be consistent. Hence, interaction consistency aims at guaranteeing global properties, such as the absence of dead non-final states and the reachability of the interaction goal. Consistency for process models that abstract a common business process, in turn,

aims at ensuring that there are no behavioural deviations for corresponding activities. As a consequence, our consistency criteria would not allow for parallelisation or sequentialisation of activities that is visible in the behavioural abstraction.

6.5 CONCLUSION

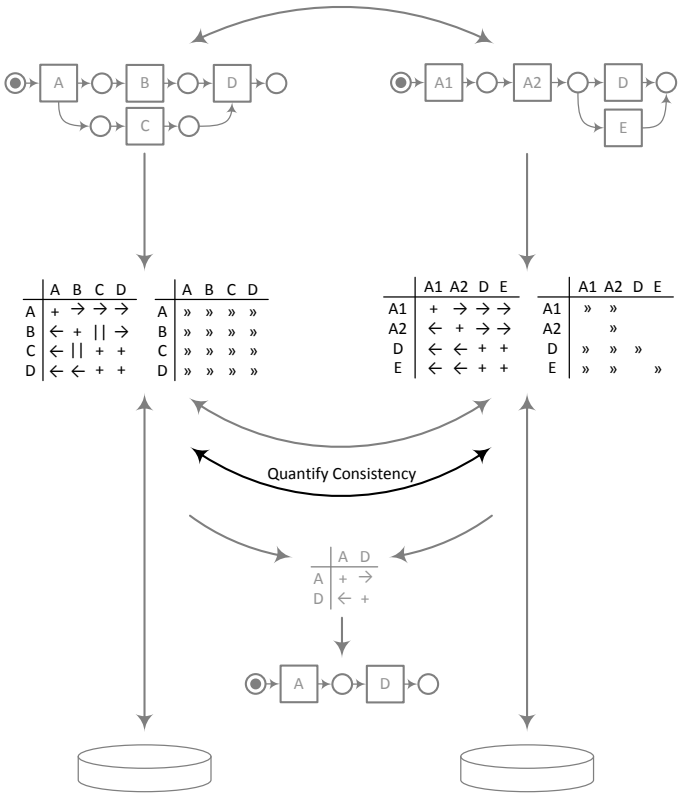
In this chapter, we focussed on the problem of deciding behaviour consistency for two process models that are aligned by correspondences. We discussed two different approaches to assess behaviour consistency, based on behaviour equivalences or using behavioural relations. For the latter option, we motivated the application of relations that capture indirect dependencies with the existence of partial alignments. We proposed different consistency notions that leverage behavioural profiles and span a spectrum of consistency criteria. Then, we focussed on the application of our consistency criteria for the evaluation of consistency between business centred process models and technical workflow models. We presented findings from an experiment on the consistency perception of process modelling experts. For this use case, the weak criteria of our consistency spectrum show a good approximation of the human consistency perception. This provides us with evidence that the proposed notions are meaningful to modelling experts.

Behavioural profiles are a behavioural abstraction. On the one hand, behavioural profiles can be computed efficiently for a broad class of process models. On the other hand, the application of a behavioural abstraction raises the question whether the neglected behavioural aspects are crucial for deciding behaviour consistency. For the investigated use case, this does not seem to be the case. It is remarkable that the most abstract criterion showed by far the best approximation of human consistency perception. Again, we highlight that the abstraction of behaviour is well-motivated. To conclude, our experimental results indicate that the order of potential activity occurrence as captured by the behavioural profile is essential for accessing behaviour consistency.

7

QUANTIFYING PROCESS MODEL CONSISTENCY

This chapter is based on results published in [505, 499].



So far, we have addressed analysis of behaviour consistency between aligned process models with Boolean consistency criteria. In many cases, however, slight deviation even from rather weak consistency criteria based on behavioural profiles can be expected to hold in practise. Following the line of the consistency notions presented in the previous chapter, this chapter introduces consistency measures. Those allow for quantification of behaviour consistency of an alignment between process models. As in the previous chapter, we first discuss behaviour equivalences and behavioural relations as different formal groundings for consistency quantification in Section 7.1. Then, Section 7.2 defines consistency measures based on behavioural profiles. We also discuss their properties and ways to consider confidence values of correspondences during measurement. Section 7.3 presents an experimental evaluation of the consistency

measures, in which we assess consistency between process models of a reference model. Then, we turn the focus on changes of process models. Those are likely to affect behaviour consistency between aligned process models. The question of how to support behaviour consistent propagation of changes is addressed in Section 7.4. Finally, we review related work in Section 7.5 and conclude the chapter in Section 7.6.

7.1 CONSISTENCY QUANTIFICATION

To illustrate the need to quantify consistency we first take up the example of a lead-to-order process introduced in the previous chapter. Following on the discussion of Boolean consistency criteria, we elaborate on how to quantify consistency based on behaviour equivalences and based on behavioural relations.

An Example Setting

For illustration purposes, we revisit the lead-to-order process introduced in Section 6.1. Figure 47 depicts two net systems that capture this process and are aligned by correspondences. Again, we see that the overall processing of a lead is similar according to both systems. In contrast to the examples discussed in Section 6.1, the two systems in Figure 47 do not satisfy the weakest consistency criterion based on behavioural profiles. We observe different orders of potential occurrence for pairs of corresponding transitions. That is, transition 'Analyse Competitors' and 'Submit Quote' are in strict order in system (a), whereas their counterparts are in interleaving order in system (b). There may be different reasons for this deviation. On the one hand, the interleaving order in the lower system may be explained by the compound activity represented by transition 'Enter & Send Quote'. One may imagine that a system allows for entering quote details first before the counter-offers are evaluated and the quote is finally sent. Such a processing may be approximated by the concurrent enabling of the transitions once entering the details and sending the quote are represented by one transition. On the other hand, the history of counter-offers may be relevant not for the quote submission, but only for the negotiation phase in the lower system.

Regardless of the reason for this deviation, the behavioural difference violates all of the proposed Boolean consistency criteria. Arguably, the deviation is minor and the overall processing is similar. Consistency measures are required to judge on the extent of the behavioural deviation. Then, pairs of aligned process models that show largely consistent behaviour are differentiated from those that have virtually nothing in common.

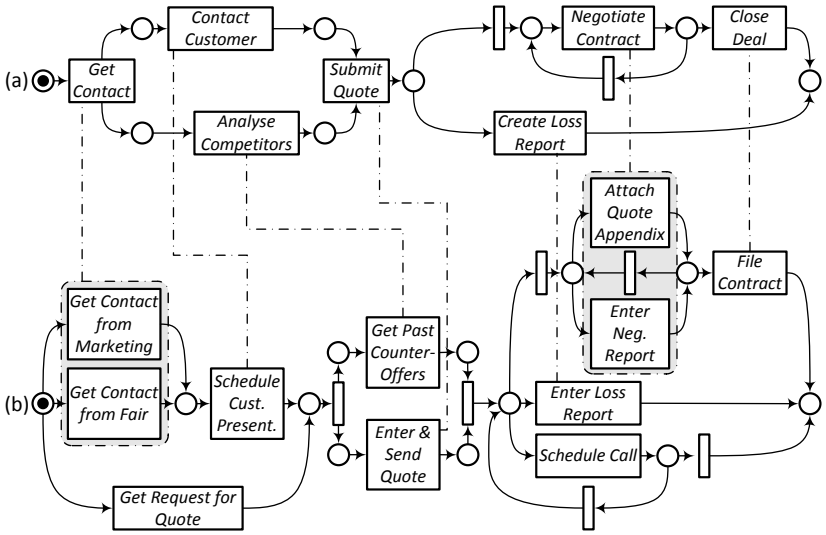


Figure 47: Two net systems depicting a lead-to-order process that are aligned by correspondences.

Consistency Quantification based on Behaviour Equivalences

Consistency of aligned process models can be quantified based on behaviour equivalences. Taking one of the notions of the linear time – branching time spectrum, there are different ways to capture the extent to which an equivalence notion is satisfied.

Following on the idea behind trace equivalence, the ratio of shared (completed) traces and all (completed) traces of two net systems may be applied as a measure if the sets of traces are finite [107, 136]. This, again, raises the question of how to cope with activities that are not aligned and with complex correspondences. In principle, the (partial) solutions outlined in Section 6.1 can be applied, even though their operationalisation is challenging. For instance, the decision to either block or hide activities that are not aligned affects the measures. Hence, measuring behaviour consistency turns into an optimisation problem. The degree of consistency depends on an optimal combination of blocking and hiding activities. Our example in Figure 47 illustrates that both options have to be considered, ignoring that sets of traces are not even finite. Assume that we neglect the circuits by blocking the respective transitions without labels in both systems. Then, transitions 'Get Request for Quote' and 'Schedule Call' in system (b) need to be blocked and all remaining transitions without labels in both systems need to be hidden to obtain a maximal consistency value.

Following on simulation equivalence, a different approach has been proposed in [427]. Given two state spaces, simulation decides whether each state transition in one state space can be mirrored in the other state space. This notion may be adap-

ted, such that instead of mirroring a state transition, a stuttering transition may be performed [427]. Taking such a stuttering step is penalised based on a notion of similarity of state transitions. This allows for conducting a *weighted quantitative simulation*. It quantifies to which extend one state space simulates another state space. Although this technique can cope with transitions that are not aligned, it remains open how to deal with complex correspondences. Quantification of the degree to which two state spaces simulate each other was also addressed in [330]. This work, again, leverages a similarity score for state transitions. Then, the similarity of states is evaluated iteratively by considering the similarities of neighbouring state transitions and states. The approach terminates after a fixpoint or an iteration boundary is reached. However, the treatment of complex correspondences is not addressed.

These approaches illustrate that quantification of similar behaviour may be directly grounded on behaviour equivalences. Nevertheless, the operationalisation of such approaches in the context of partial alignments that comprise complex correspondences is challenging and only partially addressed in the literature. Further, all of these approaches can be assumed to be computationally hard in the general case as they exploit the sets of (completed) traces or the state spaces, respectively.

Consistency Quantification based on Behavioural Relations

Behavioural relations allow for an alternative way to quantify consistency of aligned process models. In the previous chapter, we discussed consistency criteria that require all behavioural relations between pairs of transitions in one system to be consistent with the relations for pairs of corresponding transitions in the other system. Different behavioural semantics qualify for being the basis of these consistency criteria. We relied on behavioural profiles and the type equivalence of profile relations. This choice was motivated by the fact that indirect dependencies are insensitive to extensions.

With the relations of the behavioural profile, consistency of aligned process models can be quantified in a straight-forward manner. As for the Boolean consistency criteria, we check for all behavioural relations between pairs of transitions whether they are type equivalent to the relations between pairs of corresponding transitions. The ratio of transition pairs that show type equivalent relations to all aligned transition pairs provides us with a measure of consistency.

Referring to the alignment depicted in Figure 47, we will derive a rather high consistency value. Most of the behavioural relations between transition pairs in system (a) are type equival-

ent to the relations between pairs of corresponding transitions in system (b). However, the aforementioned deviation, strict order between transitions 'Analyse Competitors' and 'Submit Quote' in system (a) but interleaving for their counterparts in system (b), will lead to a consistency value below one.

7.2 CONSISTENCY MEASURES

This section is dedicated to the formalisation of our approach to consistency quantification. First, we introduce consistency measures. Second, we discuss their properties. Third, we elaborate on how the confidence values of correspondences may be taken into account when measuring behaviour consistency.

Definitions of Consistency Measures

In Section 6.2, we introduced a spectrum of consistency criteria based on behavioural profiles. This spectrum is spanned by three criteria, weak behavioural profile consistency, behavioural profile consistency, and causal behavioural profile consistency. The criteria differ with respect to the considered behavioural relations. In this line, we also define three consistency measures.

We first define the set of behavioural profile consistent transition pairs. Given an alignment between two net systems, this set contains all pairs of transitions that are part of a correspondence and for which the corresponding transitions show type equivalent profile relations. Using the notion of aligned transitions introduced in Definition 6.2.1, we define this set as follows.

Definition 7.2.1 (BP Consistent Transition Pairs)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2, \}$ their behavioural profiles, and $\sim \subseteq T_1 \times T_2$ a correspondence relation. Let $R_1 \in \mathcal{B}_1 \cup \{\rightsquigarrow_1^{-1}\}$ and $R_2 \in \mathcal{B}_2 \cup \{\rightsquigarrow_2^{-1}\}$. The set of *behavioural profile consistent transition pairs* $CT_1^{\sim} \subseteq (T_1^{\sim} \times T_1^{\sim})$ for S_1 contains all pairs (t_x, t_y), such that

- if $t_x = t_y$, then $\forall\ t_s \in T_2^{\sim}$ with $t_x \sim t_s$ it holds $(t_x R_1 t_x \wedge t_s R_2 t_s) \Rightarrow R_1 \simeq R_2$,
- if $t_x \neq t_y$, then $\forall\ t_s, t_t \in T_2^{\sim}$ with $t_s \neq t_t$, $t_x \sim t_s$, and $t_y \sim t_t$ it holds either (1) $(t_x R_1 t_y \wedge t_s R_2 t_t) \Rightarrow R_1 \simeq R_2$ or (2) $t_x \sim t_t$ and $t_y \sim t_s$.

The set $CT_2^{\sim}$ for S_2 is defined analogously.

As for the Boolean consistency criteria introduced in Section 6.2, the set of behavioural profile consistent transition pairs considers the profile relations only for pairs of transitions that relate to different correspondences. Further, transition pairs that are part of

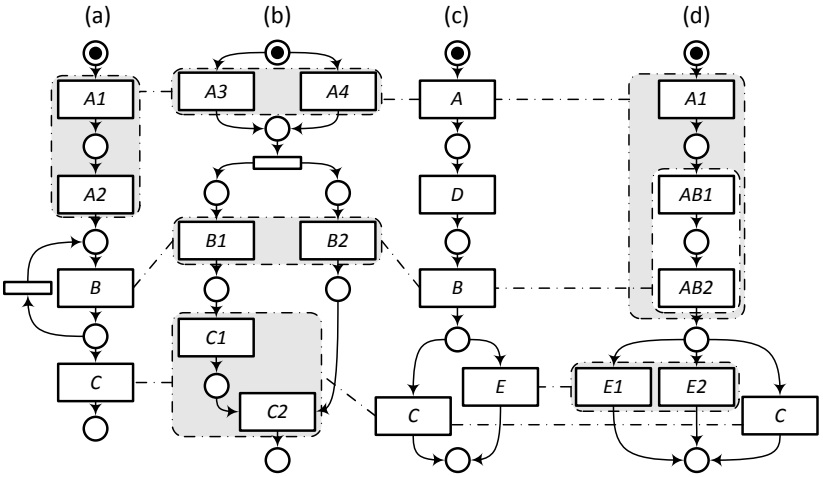


Figure 48: Aligned net systems to illustrate the consistency measurement.

an overlap of two correspondences are not taken into account either.

Using the definition of behavioural profile consistent transition pairs, we are able to quantify the consistency of an alignment between two net systems. The main idea is to determine the share of transition pairs, for which the corresponding transitions have type equivalent behavioural relations. As the alignment may comprise complex correspondences with different cardinalities, our measures are based on consistent transition pairs of both aligned net systems.

First, we define the degree of weak behavioural profile consistency. It leverages the relations of the behavioural profile except for the self-relations of transitions.

Definition 7.2.2 (Degree of Weak BP Consistency)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, and $\sim \subseteq T_1 \times T_2$ a correspondence relation. The *degree of weak behavioural profile consistency* of $\sim$ is defined as

$$\mathcal{WPC}^{\sim} = \frac{\left|CT_1^{\sim} \setminus id_{T_1^{\sim}}\right| + \left|CT_2^{\sim} \setminus id_{T_2^{\sim}}\right|}{\left|(T_1^{\sim} \times T_1^{\sim}) \setminus id_{T_1^{\sim}}\right| + \left|(T_2^{\sim} \times T_2^{\sim}) \setminus id_{T_2^{\sim}}\right|}.$$

We illustrate this degree with Figure 48. For the alignment between systems (a) and (b), we observe different behavioural relations for the aligned transitions, even if self-relations are neglected. The order of potential occurrence between transitions B and C in system (a) is not mirrored in system (b), as it holds C1||B2. Besides this deviation, most relations are type equivalent for the respective pairs of transitions, so that we obtain a degree of weak behavioural profile consistency of $\mathcal{WPC} = \frac{10+28}{12+30} \approx 0.90$. The relation between transitions C1 and B2 in system (b) is also

inconsistent regarding the alignment with system (c). As system (c) contains only three transitions that have counterparts in system (b), the normalisation differs compared to the first alignment. For the alignment between systems (b) and (c), we obtain a consistency value of $\mathcal{WPC} = \frac{28+4}{30+6} \approx 0.89$. The alignment between systems (c) and (d) turns out to satisfy weak behavioural profile consistency as introduced in the previous chapter, see Section 6.2. This is also reflected in the degree of weak behavioural profile consistency, for which we obtain a value of $\mathcal{WPC} = \frac{12+30}{12+30} = 1$.

Quantification of consistency may also be done in the line of the behavioural profile consistency criterion. Then, self-relations also influence the consistency measure.

Definition 7.2.3 (Degree of BP Consistency)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, and $\sim \subseteq T_1 \times T_2$ a correspondence relation. The *degree of behavioural profile consistency* of $\sim$ is defined as

$$\mathcal{PC}^{\sim} = \frac{\left|CT_1^{\sim}\right| + \left|CT_2^{\sim}\right|}{\left|T_1^{\sim} \times T_1^{\sim}\right| + \left|T_2^{\sim} \times T_2^{\sim}\right|}.$$

We base the degree of behavioural profile consistency on the Cartesian products of aligned transitions of both systems. As a consequence, there is an implicit weighting of behavioural dependencies. A pair of equal transitions is considered once. In contrast, for any pair of distinct transitions, the reverse pair also influences the consistency measure. If this would turn out to be inappropriate in a certain context, however, our measures may easily be adapted.

For the examples in Figure 48, we obtain the following results. Consideration of self-relations affects the consistency measurement for the alignment between systems (a) and (b). The potential repetition of transition B in system (a), B||B, is not respected by both corresponding transitions B1 and B2 in system (b). For this alignment, we obtain a value of $\mathcal{PC} = \frac{13+32}{16+36} \approx 0.87$. There are no deviations regarding self-relations for the alignments between systems (b) and (c). This leads to a slightly higher consistency value compared to the degree of weak behavioural profile consistency, that is $\mathcal{PC} = \frac{34+7}{36+9} \approx 0.91$. For the alignment between systems (c) and (d), again, we derive a value of $\mathcal{PC} = \frac{16+36}{16+36} = 1$.

Finally, quantification may be based on all relations of the causal behavioural profile. To take co-occurrences into account, we lift the notion of consistent transition pairs to causal behavioural profiles. We capture all pairs of aligned transitions, for which the corresponding transitions show type equivalent profile relations and consistent co-occurrences.

Definition 7.2.4 (CBP Consistent Transition Pairs)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, $\mathcal{CB}_1 = \{\rightsquigarrow_1, +_1, \|_1, \gg_1\}$ and $\mathcal{CB}_2 = \{\rightsquigarrow_2, +_2, \|_2, \gg_2\}$ their causal behavioural profiles, and $\sim \subseteq T_1 \times T_2$ a correspondence relation. Let $CT_1^{\sim}$ be the set of behavioural profile consistent transition pairs for S_1. The set of *causal behavioural profile consistent transition pairs* $CCT_1^{\sim} \subseteq CT_1^{\sim}$ for S_1 contains all pairs (t_x, t_y), such that if $t_x \neq t_y$ then for all transitions $t_s, t_t \in T_2^{\sim}$ with $t_s \neq t_t$, $t_x \sim t_s$, and $t_y \sim t_t$ it holds either (1) $t_x \gg_1 t_y \Leftrightarrow t_s \gg_2 t_t$ or (2) $t_x \sim t_t$ and $t_y \sim t_s$. The set $CCT_2^{\sim}$ for S_2 is defined analogously.

We define the degree of causal behavioural profile consistency as follows.

Definition 7.2.5 (Degree of CBP Consistency)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, and $\sim \subseteq T_1 \times T_2$ a correspondence relation. The *degree of causal behavioural profile consistency* of $\sim$ is defined as

$$\mathcal{CPC}^{\sim} = \frac{\left|CCT_1^{\sim}\right| + \left|CCT_2^{\sim}\right|}{\left|T_1^{\sim} \times T_1^{\sim}\right| + \left|T_2^{\sim} \times T_2^{\sim}\right|}.$$

Again, we refer to Figure 48 for illustration purposes. For the alignment between systems (a) and (b), several co-occurrences present in system (a) are not mirrored for corresponding transitions in system (b). This is mainly caused by the application of the binary co-occurrence relation, which provides an abstracted view on the behaviour, see also the discussion in Section 6.1. For instance, transitions A1 and B are co-occurring in both directions in system (a), $A1 \gg B$ and $B \gg A1$. In system (b), co-occurrence holds only for one direction as $A1 \gg B1$, $A2 \gg B1$, $A1 \gg B2$ and $A2 \gg B2$, but $B1 \not\gg A1$, $B1 \not\gg A2$, $B2 \not\gg A1$, and $B2 \not\gg A2$. For this alignment, we obtain a degree of causal behavioural profile consistency of $\mathcal{CPC} = \frac{9+24}{16+36} \approx 0.63$. Similar observations are made for the alignment between systems (b) and (c). The co-occurrence dependency $B \gg A$ in system (c) is not mirrored in system (b). In addition, transitions $\{B1, B2, C1, C2\}$ are all pairwise co-occurring in system (b). These dependencies are only partly satisfied in system (c), as it holds $C \gg B$ but $B \not\gg C$. This affects the degree of causal behavioural profile consistency, for which we obtain a value of $\mathcal{CPC} = \frac{19+5}{36+9} \approx 0.53$. The alignment between systems (c) and (d) satisfies even the strongest consistency criteria introduced in Section 6.2, i. e., causal behavioural profile consistency. Hence, we obtain a degree of causal behavioural profile consistency of $\mathcal{CPC} = \frac{16+36}{16+36} = 1$.

Properties of Consistency Measures

All three degrees to quantify consistency range between one and zero. A degree of one guarantees that all dependencies are equal for the aligned transitions of two net systems. The set of dependencies that influences the measure, in turn, is determined by the applied behavioural abstraction.

Our measures quantify the quality of the correspondence relation of an alignment. Hence, their definition is independent of the coverage of the net system by the correspondence relation. The ratio of transitions that are aligned in both systems to all transitions does not affect the measures. As a consequence, removing a correspondence from an alignment may increase the degree of consistency. The motivation for this operationalisation directly follows from our use case. Process models created for different purposes are likely to be partially overlapping in coverage of business operations. Certain parts of operations are of relevance only in a certain context. Thus, it will rarely be the case that the correspondence relation is total from one model to the other model.

Our focus on the quality of the correspondence relation has to be seen as a major difference with respect to measure of process model similarity. Those typically quantify the size of the overlap in terms of similar activities of two process models. We further discuss approaches to similarity measurement when reviewing related work.

Despite these conceptual differences, our measures show certain properties that are closely related to the mathematical notions of a similarity and a metric. According to [153], a similarity satisfies the positiveness, maximality, and symmetry properties. Positiveness and symmetry relate to the comparison of two net systems and are satisfied by our measures.

Property 7.2.1. Given two net systems aligned by correspondences, the degrees of weak behavioural profile consistency, behavioural profile consistency, and causal behavioural profile consistency are positive.

For all three degrees, the respective sets of consistent transitions referenced in the numerators are subsets of the Cartesian products of the aligned transitions referenced in the denominators. For a correspondence relation $\sim \subseteq T_1 \times T_2$, it holds $CT_1^{\sim}, CCT_1^{\sim} \subseteq T_1^{\sim}$ and $CT_2^{\sim}, CCT_2^{\sim} \subseteq T_2^{\sim}$. Hence, all degrees yield a positive value.

Property 7.2.2. Given two net systems aligned by correspondences, the degrees of weak behavioural profile consistency, behavioural profile consistency, and causal behavioural profile consistency are symmetric.

Symmetry follows directly from the symmetry of the summation operator applied for the set cardinalities in the numerators and denominators of the degree computation.

The maximality property of a similarity states that the maximal value is assumed when comparing an entity with itself. This property is not applicable in our context. It would require a notion of an implicit self-alignment for net systems.

Closely related to similarities are distance metrics. A distance metric, or dissimilarity, is often interpreted as the dual operation for similarity, defined as the similarity subtracted from one. Our measures cannot be used as metrics for the comparison of aligned net systems. In particular, any dissimilarity that is grounded on our measures does not satisfy the triangle inequality. This property allows for concluding the minimal and maximal distance between two entities if their pairwise distance to a third entity is known. The presented degrees are normalised by the size of the alignment, i. e., the number of transitions that are part of correspondences. Hence, our measures are independent of the size of the respective net systems. That precludes any possibility of using them for the definition of distance metrics for the comparison of net systems.

Confidence of Consistency Measures

The consistency measures introduces earlier, treat all aligned transitions along with the behavioural relations between them equally. All correspondences are equally important when evaluating the consistency. In the previous chapter, we discussed how confidence values for correspondences are interpreted against the background of Boolean consistency notions. Any decision taken based on such a consistency notion has a confidence that is computed as the average of the confidence values for all elementary correspondences. In principle, this approach may also be taken for the consistency measures. The degree of consistency derived by one of the proposed measures is then associated with the confidence that the correspondences between two net systems hold.

Once consistency is quantified, however, the confidence of correspondences may also influence the measures directly. It can be used to assign a weighting to correspondences. The motivation behind is that equality of behavioural relations for corresponding transitions is more important if we are highly confident that the respective correspondences actually hold between both systems. On the other hand, violation of behavioural relations is less harmful if there is a high uncertainty whether the respective correspondences have been identified correctly.

Following this idea, we provide three weighted variants of the proposed consistency measures. As an auxiliary notion, we first define the average confidence function. For a single aligned transition, this function aggregates the confidence values of elementary correspondences that refer to this transition by the arithmetic mean. Again, we rely on the notion of aligned transitions introduced in Definition 6.2.1.

Definition 7.2.6 (Average Confidence Function)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, and $(\sim, \zeta)$ an alignment with $\sim \subseteq T_1 \times T_2$. The *average confidence function* $\bar{\zeta} : (T_1^\sim \cup T_2^\sim) \mapsto [0, 1]$ is defined for a transition $t \in (T_1^\sim \cup T_2^\sim)$ as

$$\bar{\zeta}(t) = \frac{\sum_{(t_x,t_s)\in \sim \wedge\ (t=t_x\ \vee\ t=t_s)} \zeta(t_x, t_s)}{|\{(t_x, t_s) \in \sim \mid t = t_x \ \vee \ t = t_s\}|}.$$

With the average confidence function, we adapt the three aforementioned degrees of consistency.

Definition 7.2.7 (Weighted Degrees of Consistency)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, and $\sim \subseteq T_1 \times T_2$ a correspondence relation.
The *weighted degree of weak behavioural profile consistency* of $\sim$ is

$$\mathcal{WPC}_\zeta^\sim = \frac{\sum_{(t_x,t_y)\in(CT_1^\sim \setminus id_{T_1^\sim})}(\bar{\zeta}(t_x)+\bar{\zeta}(t_y))+\sum_{(t_s,t_t)\in(CT_2^\sim \setminus id_{T_2^\sim})}(\bar{\zeta}(t_s)+\bar{\zeta}(t_t))}{\sum_{(t_x,t_y)\in((T_1^\sim \times T_1^\sim)\setminus id_{T_1^\sim})}(\bar{\zeta}(t_x)+\bar{\zeta}(t_y))+\sum_{(t_s,t_t)\in((T_2^\sim \times T_2^\sim)\setminus id_{T_2^\sim})}(\bar{\zeta}(t_s)+\bar{\zeta}(t_t))}.$$

The *weighted degree of behavioural profile consistency* of $\sim$ is

$$\mathcal{PC}_\zeta^\sim = \frac{\sum_{(t_x,t_y)\in CT_1^\sim}(\bar{\zeta}(t_x)+\bar{\zeta}(t_y))+\sum_{(t_s,t_t)\in CT_2^\sim}(\bar{\zeta}(t_s)+\bar{\zeta}(t_t))}{\sum_{(t_x,t_y)\in(T_1^\sim \times T_1^\sim)}(\bar{\zeta}(t_x)+\bar{\zeta}(t_y))+\sum_{(t_s,t_t)\in(T_2^\sim \times T_2^\sim)}(\bar{\zeta}(t_s)+\bar{\zeta}(t_t))}.$$

The *weighted degree of causal behavioural profile consistency* of $\sim$ is

$$\mathcal{CPC}_\zeta^\sim = \frac{\sum_{(t_x,t_y)\in CCT_1^\sim}(\bar{\zeta}(t_x)+\bar{\zeta}(t_y))+\sum_{(t_s,t_t)\in CCT_2^\sim}(\bar{\zeta}(t_s)+\bar{\zeta}(t_t))}{\sum_{(t_x,t_y)\in(T_1^\sim \times T_1^\sim)}(\bar{\zeta}(t_x)+\bar{\zeta}(t_y))+\sum_{(t_s,t_t)\in(T_2^\sim \times T_2^\sim)}(\bar{\zeta}(t_s)+\bar{\zeta}(t_t))}.$$

We illustrate the weighted measures with the example alignment depicted in Figure 49. We showed both net systems already in Figure 48, when discussing the non-weighted consistency measures. For the depicted alignment, we obtained consistency values of $\mathcal{WPC} \approx 0.90$, $\mathcal{PC} \approx 0.87$, and $\mathcal{CPC} \approx 0.63$. Figure 49 now illustrates confidence values for the correspondences. For simplicity, we assume all elementary correspondences that form a complex correspondence to have equal confidence values. Apparently, the correspondence between transition B in system (a)

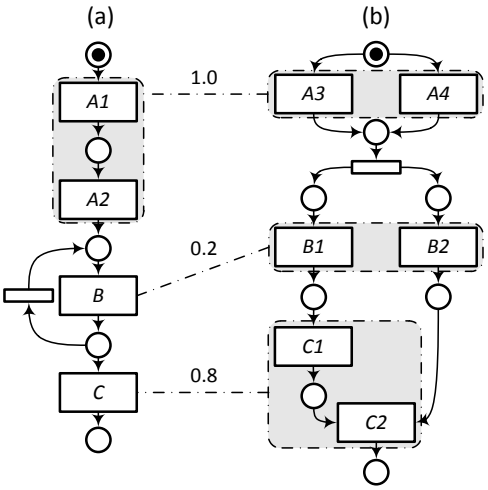


Figure 49: Aligned net systems with confidence values to illustrate the weighted consistency measurement.

and transitions {B1, B2} in system (b) shows a rather high uncertainty. Even though this correspondence may be removed from the alignment due to the low confidence value, we rely on this example for illustration purposes. Application of the weighted consistency measures yields the following results, $\mathcal{WPC} \approx 0.93$, $\mathcal{PC} \approx 0.93$, and $\mathcal{CPC} \approx 0.69$. All values are above those obtained with the non-weighted measures. Most behavioural issues of the alignment relate to transition B in system (a) and transitions {B1, B2} in system (b). As the correspondence between them has a low confidence value, all behavioural relations that refer to one of these transitions have a lower influence on the consistency evaluation compared to those that refer to other transitions.

7.3 EXPERIMENTAL EVALUATION

We evaluated the introduced consistency measures with an experimental setup. Our experiment incorporated the models of the SAP reference model [95]. We already used this model collection to evaluate the different approaches to the computation of behavioural profiles in Section 5.4. The SAP reference model describes the functionality of the SAP R/3 system and captures different functional aspects of an enterprise, such as sales or accounting. However, the models are not orthogonal. There are rather large clusters of models that show a functional overlap, manifested in equally labelled functions and events in the EPCs. We can only speculate on the reasons for this observation. On the one hand, the models may refer to different variations of a business process, i. e., different originals. On the other hand, they may abstract a common process but assume the perspective of a certain department on this process. As such, the process models

of this collection may not be a typical example for consistency analysis as mainly investigated in this work. Nevertheless, an evaluation of behaviour consistency provides insights on how well the models in this collection are aligned.

Our experiment on behaviour consistency between models of the SAP reference model focussed on the influence of causal dependencies. In previous chapter, we discussed that such dependencies may be broken once only the *happy path* of business operations is captured. The behavioural profile focusses on the order of potential occurrence and, therefore, explicitly neglects these aspects. To investigate the influence of causal dependencies, our experiment relied on two measures. As a baseline, we used a measure that builds upon completed trace equivalence and the idea of projection inheritance. It is based on completed trace semantics of net systems and incorporates projection to cope with transitions that are not aligned. As we will discuss later, considering only completed traces instead of all traces appears to be particularly suited if transitions are projected. In addition, we used the degree of behavioural profile consistency that neglects causal dependencies. Note that for the net systems that we evaluated, behavioural profile consistency coincided with weak behavioural profile consistency. We considered only sound acyclic net systems to be able to exploit the sets of completed traces for our baseline measure.

The remainder of this section is structured as follows. First, we formally define the baseline measure. Then, we summarise the experimental setup and present the obtained consistency results. Finally, we discuss the results with selected examples from the model collection.

A Baseline Measure based on Projected Completed Trace Equivalence

Our baseline measure follows on completed trace equivalence. Hence, we assume that a net system has a finite set of traces. Completed trace equivalence requires two net systems to have equal sets of completed traces. Those are characterised by firing sequences that start in the initial marking of the net system and end in a dead marking.

Definition 7.3.1 (Completed Traces)
Let $S = (N, M_0)$ be a net system with $N_1 = (P, T, F)$ that has a finite set of traces $\mathcal{T}$. The set of *completed traces* $\mathcal{CT} \subseteq \mathcal{T}$ contains a firing sequence $\sigma \in \mathcal{T}$ if the marking M_1 reached by σ, $(N, M_0)[\sigma\rangle(N, M_1)$, is a dead marking.

To cope with transitions that are not aligned, we adapt the idea of projection inheritance [451, 33]. That is, we hide these transitions from the traces. We decided to adapt projection inherit-

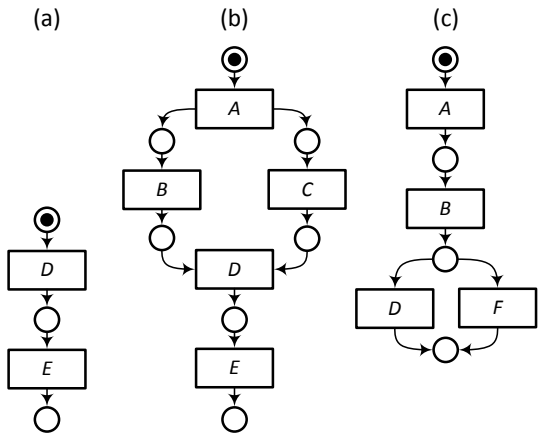


Figure 50: Aligned net systems (elementary correspondences according to transition labels) to illustrate the consistency measurement based on projected completed traces.

ance as it lifts the completed trace equivalence criterion directly to alignments of process models. In contrast to protocol inheritance [451, 33], it considers all execution paths.

Given a correspondence relation, we define how projection is applied to a firing sequence of one of the aligned net systems. The following definition uses the notion of aligned transitions, see Definition 6.2.1.

Definition 7.3.2 (Complete Trace Projection)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$ that have finite sets of completed traces $\mathcal{CT}_1$ and $\mathcal{CT}_2$. Let $\sim \subseteq T_1 \times T_2$ a correspondence relation.

- For a firing sequence $\sigma \in \mathcal{CT}_1$ of length n, the set of *aligned transitions up to index* $j \in \mathbb{N}$, $1 \leqslant j \leqslant n$, is defined as $T^{\sim}_{\sigma|j} = \{t_x \in \sigma \mid x < j \wedge t_x \in T_1^{\sim}\}$.
- For a firing sequence $\sigma \in \mathcal{CT}_1$ of length n, the *projection function* $\tau^{\sim}$ returns the projected firing sequence that contains all aligned transitions of σ, i. e., $\tau^{\sim}(\sigma) = \bigcup_{i=0}^{|T^{\sim}_{\sigma|n}|}(i, t_i)$ with $t_i \in T_1^{\sim}$, such that $\exists\, j \in \mathbb{N}\ [\ (j, t_i) \in \sigma \wedge i = |T^{\sim}_{\sigma|j}|\ ]$.
- The set of *projected completed traces* $\mathcal{CT}_1^{\sim} = \bigcup_{\sigma \in \mathcal{CT}_1} \tau^{\sim}(\sigma)$ of S_1 comprises all projected firing sequences.

We illustrate the notion of projected firing sequences with the examples in Figure 50. Here, we assume elementary correspondences to be defined according to the transition labels. Consider the alignment between systems (a) and (b). In system (b), there is a firing sequence $\sigma = \{(0, A), (1, C), (2, B), (3, D), (4, E)\}$, also referred to as $\sigma = \langle A, C, B, D, E\rangle$, reachable from the initial marking. Transitions A, B, and C are not aligned with respect to system (a). Hence, the respective projected firing sequence is defined as $\tau(\sigma) = \{(0, D), (1, E)\}$.

For any firing sequence that does not contain any aligned transition, the projected firing sequence is the empty sequence. This observation motivates the usage of completed traces instead of relying on all traces for consistency analysis. Consider again the systems (a) and (b) in Figure 50. For the firing sequence $\sigma = \langle A, C \rangle$ in system (b), projection results in the empty sequence. Apparently, there is no firing sequence in system (a), for which projection yields the empty sequence. To avoid such anomalies, we consider only completed traces.

Using the concept of a trace projection, we can compare the sets of completed traces of two aligned net systems if the correspondence relation is functional and injective. The ratio of shared completed traces to all completed traces of two net systems is used to quantify consistency. We define the degree of trace consistency as follows.

Definition 7.3.3 (Degree of Trace Consistency)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$ that have finite sets of completed traces $\mathcal{CT}_1$ and $\mathcal{CT}_2$. Let $\sim \subseteq T_1 \times T_2$ a correspondence relation and $\mathcal{CT}_1^{\sim}$, $\mathcal{CT}_2^{\sim}$ the sets of projected completed traces.

- Two firing sequences $\sigma_1 \in \mathcal{CT}_1^{\sim}$ and $\sigma_2 \in \mathcal{CT}_2^{\sim}$ are consistent, denoted by $\sigma_1^{\sim} \simeq \sigma_2^{\sim}$, iff $\sigma_1^{\sim} = \sigma_2^{\sim} = \emptyset$ or $\forall\ t_i \in \sigma_1^{\sim}\ [\ \exists\ t_j \in \sigma_2^{\sim}\ [\ i = j\ \wedge\ t_i \sim t_j\ ]]$.
- With $\mathcal{CCT}_1 = \{\sigma_1^{\sim} \in \mathcal{CT}_1^{\sim} \mid \exists\ \sigma_2^{\sim} \in \mathcal{CT}_2^{\sim}\ [\ \sigma_1^{\sim} \simeq \sigma_2^{\sim}]\}$ and $\mathcal{CCT}_2 = \{\sigma_1^{\sim} \in \mathcal{CT}_2^{\sim} \mid \exists\ \sigma_2^{\sim} \in \mathcal{CT}_1^{\sim}\ [\ \sigma_2^{\sim} \simeq \sigma_1^{\sim}]\}$ the *degree of trace consistency* of $\sim$ is

$$\mathcal{TC}^{\sim} = \frac{|\mathcal{CCT}_1| + |\mathcal{CCT}_2|}{|\mathcal{CT}_1^{\sim}| + |\mathcal{CT}_2^{\sim}|}.$$

For the example models given in Figure 50, we obtain the following results. Consider the alignment between systems (a) and (b). For both aligned net systems, the sets of projected completed traces comprise solely a single firing sequence. As these firing sequences are consistent, this yields of degree of trace consistency of $\mathcal{TC} = \frac{1+1}{1+1} = 1$. For the alignment between systems (b) and (c), we obtain a different result. The set of projected completed traces for system (b) comprises one firing sequence $\sigma = \langle A, B, D \rangle$. For the system (c), there are two firing sequences in the set of projected completed traces, i. e., $\sigma_1 = \langle A, B, D \rangle$ and $\sigma_2 = \langle A, B \rangle$. Since the latter is without consistent counterpart in system (b), the degree of trace consistency is $\mathcal{TC} = \frac{1+1}{1+2} \approx 0.67$.

Experimental Setup

For our experiment, we extracted a subset of models of the SAP reference model. This selection is similar to the one described

Table 4: Consistency results for alignments between two process models of the SAP reference model.

Category	# Align.	Avg. $\mathcal{PC}$	$\mathcal{PC} < 1.0$	Avg. $\mathcal{TC}$	$\mathcal{TC} < 1.0$
Functions	114	0.97	7 (6.14%)	0.89	37 (32.46%)
Events	695	0.95	71 (10.22%)	0.83	324 (46.62%)
All nodes	735	0.95	83 (11.29%)	0.81	357 (48.57%)

in Section 5.4. We selected only models that are non-trivial, free of syntax errors, and have clear instantiation semantics. Further, we excluded models that comprise OR-connectors and behavioural anomalies. Our baseline measure assumes that the set of traces of a process model is finite. Hence, cyclic process models were not further considered either. This selection leads to a sample of 420 process models. We transformed these models into sound acyclic free-choice WF-systems. As part of that, we ensured that each EPC function and EPC event is represented by a single transition.

We constructed alignments for the net systems that are built from elementary correspondences between transitions representing EPC nodes with equal labels. Then, we extracted all pairs of net systems that have been aligned by at least two correspondences. For these alignments, we then computed the degrees of trace consistency and behavioural profile consistency.

For the computation of the degree of behavioural profile consistency, we relied on the techniques introduced in Section 5.1 and Section 5.2. We computed this degree within milliseconds for an alignment. In contrast, the calculation of the degree of trace consistency is computationally hard. We implemented this computation in a prototype. Our implementation reduced the size of the systems by a *fusion of series places* [329] connected by transitions that are subject to projection. With the reduced net systems, exploration of all traces works for the majority of alignments. For 18 alignments, the computation of the degree of trace consistency turned out to be intractable. These alignments are not further considered.

Experimental Results

A summary of the obtained consistency results is presented in Table 4. We distinguish three different categories of alignments. Correspondences are established either between transitions that represent only EPC functions, only EPC events, or both node types. The node type is considered in the latter category as well,

i. e., a correspondence between two transitions requires that both represent the same type of EPC node. The second column of Table 4 shows the number of pairs of net systems that are aligned by at least two correspondences. There are significantly more alignments if EPC events are taken into account. That is due to the structure of the reference model. Single EPC models contain a lot more events than functions. In addition, end events of one EPC model reappear as start events of another EPC model.

Independent of the category, the average degree of behavioural profile consistency (Avg. $\mathcal{PC}$) is rather high. This observation is further underpinned by the low numbers of alignments that are not behavioural profile consistent, i. e., that have a consistency value below one ($\mathcal{PC} < 1.0$). The degree of trace consistency (Avg. $\mathcal{TC}$), in turn, is lower than the degree of behavioural profile consistency on average. A drastic difference is visible once the shares of alignments that are fully consistent are compared for both measures. If EPC events are considered, nearly one-half of the alignments show a degree of trace consistency below one. This yields more than 300 alignments in absolute terms. To put it differently, the degree of trace consistency suggests a high number of inconsistent alignments, whereas the degree of profile consistency points to a rather high consistency.

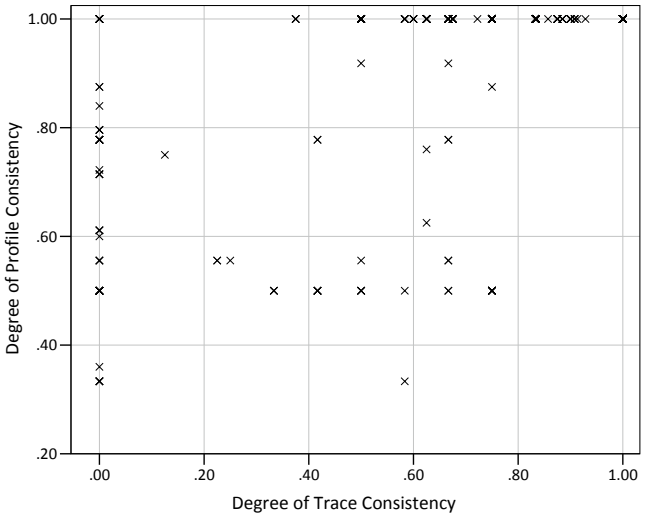
We further explore the spectrum of consistency values in Figure 51. Figure 51a depicts the distribution of behavioural profile consistency values against the degree of trace consistency.

Apparently, a degree of trace consistency of one implies that the alignment is behavioural profile consistent. For alignments that show a degree of trace consistency lower than one, we observe a spectrum of behavioural profile consistency with many different consistency values. In particular, alignments with a degree of trace consistency of zero are classified in a fine granular manner by the degree of behavioural profile consistency.

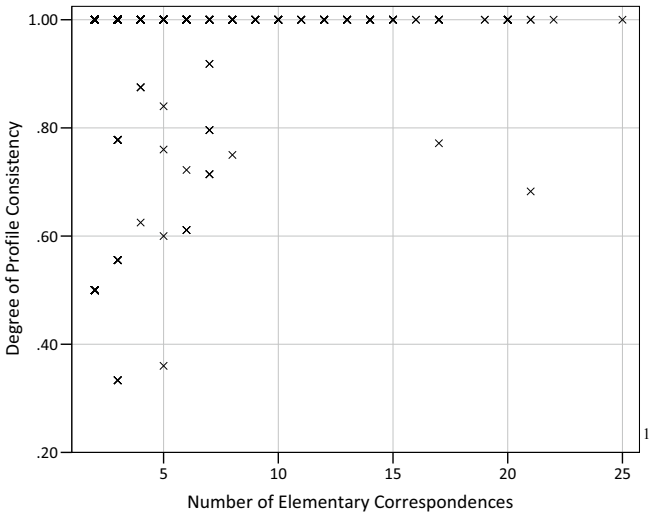
The relation between behavioural profile consistency of an alignment and the number of correspondences that form the alignment is illustrated in Figure 51b. We see that low values of behavioural profile consistency appear mainly for alignments that comprise only a few correspondences. In these cases, a single inconsistency has a bigger influence on the profile consistency value than for alignments that comprises a large number of correspondences.

Discussion of the Results

The presented results highlight that both measures may be used to quantify behaviour consistency. On the other hand, we observe a significant difference between the measure based on projected completed traces and the one based on behavioural pro-



(a)



(b)

Figure 51: Consistency values observed for alignments between two process models of the SAP reference model.

files. Conceptually, both measures differ with respect to their treatment of causal dependencies. We further investigate this aspect with two exemplary alignments from the SAP reference model.

The first example illustrates the need to judge on behaviour consistency to evaluate the quality of an alignment. Figure 52 depicts (parts of) two EPC process models from the SAP reference model. Both models specify a shipping procedure. The model in Figure 52a is part of the procurement handling section of the reference model. The model in Figure 52b originates from the sales and distribution processing. We cannot assess whether both models represent different views on a common business process or whether they capture different variations

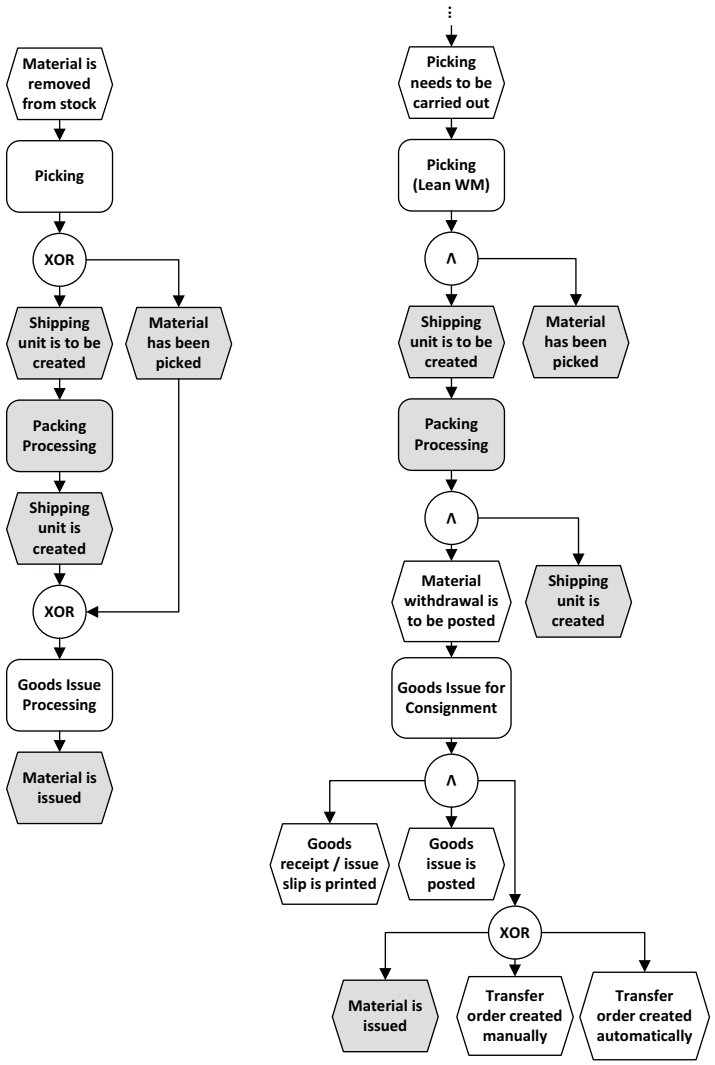


Figure 52: EPCs from the SAP reference model that are overlapping, but neither trace consistent nor behavioural profile consistent.

of business operations. Nevertheless, both models show a notable overlap and specify how material shipping is implemented. This is reflected by several nodes with identical labels. Other elements show similar labels, e. g., 'picking' and 'picking (lean WM)'. Even though these elements are not considered in our consistency analysis, they indicate high similarity of semantics.

An analysis of execution semantics of both models reveals serious differences. For instance, if the process has reached the state 'material has been picked', the model in Figure 52a specifies that after conducting the 'goods issue processing', state 'material is issued' is assumed directly. In contrast, the model in Figure 52b defines that the state 'material has been picked' implies the need for 'packing processing'. What is an exclusive choice of states in

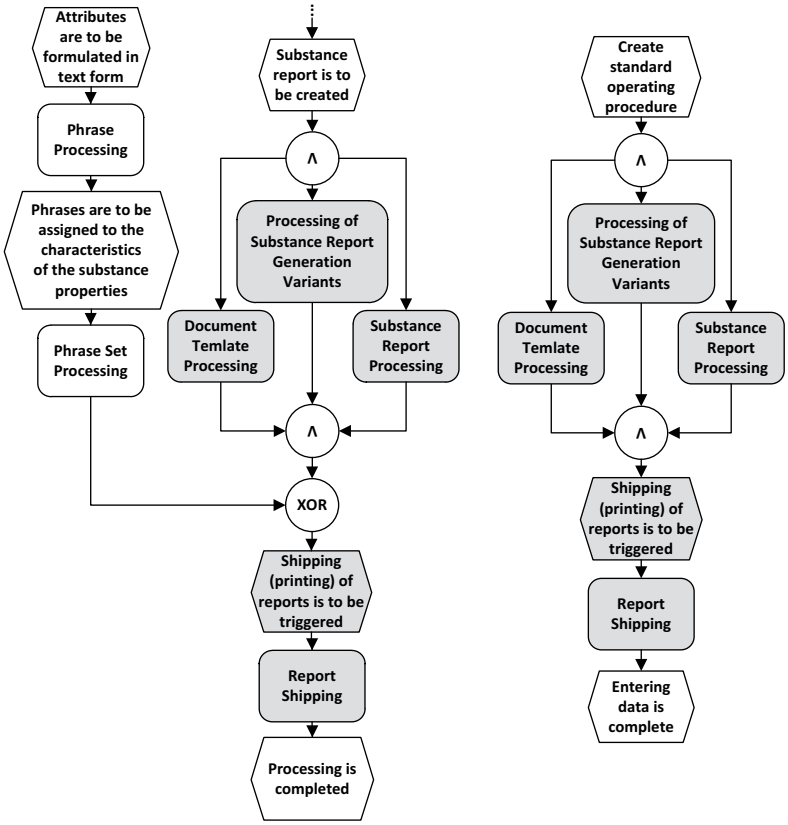


Figure 53: EPCs from the SAP reference model that are behavioural profile consistent, but not trace consistent.

the one model, corresponds to a conjunction of states in the other model. Although the 'picking' functions are not aligned, that has to be regarded as an inconsistency. Therefore, consistency of process models requires more than simply deriving the correspondences between model elements. Analysis of behaviour consistency has to be conducted. Both measures used in our experiment reveal such inconsistencies. For the models depicted in Figure 52, we obtain a degree of trace consistency of $\mathcal{TC} = 0$ and a degree of behavioural profile consistency of $\mathcal{PC} = 0.6$.

The second example focusses on the implications of the conceptual differences of the measures. Figure 53 depicts two EPC process models from the environment, health, and safety category of the reference model. Both describe the creation of incident reports and show several nodes with identical labels. The model in Figure 53a goes beyond the procedure that is depicted in the model in Figure 53b. It describes an alternative way of creating the report, i. e., it extends the model in Figure 53b. This may be caused by a focus on the most common execution path in Figure 53b. To provide an overview of the processing the al-

ternative branches may be of minor importance and, therefore, have been abstracted.

The difference between both models affects the degree of trace consistency. Several causal dependencies defined between nodes in Figure 53b are not observed for the corresponding nodes in Figure 53a. For instance, function 'report shipping' is always preceded by function 'document template processing' in the model in Figure 53b. This is not the case for the model in Figure 53a. The degree of trace consistency for this example is $\mathcal{TC} = 0.93$. Although this value indicates only minor deviations, consideration of the respective causal dependencies lowers the consistency value. In contrast, there is no difference in the potential order of execution as defined by the behavioural profile for the corresponding nodes of both models. For instance, if function 'document template processing' and function 'report shipping' are part of the process execution, the former will always happen before the latter. The strict order between the two functions holds in both models. For the models in Figure 53, we obtain a degree of behavioural profile consistency of $\mathcal{PC} = 1$, i. e., both models are behavioural profile consistent.

To conclude, the first example in Figure 52 illustrates that behaviour consistency of an alignment has to be evaluated. Pure identification of corresponding elements may lead to an alignment between process models that define contradictory behaviour. The second example in Figure 53 highlights that focussing on the *happy path* of a process may introduce causal dependencies that are not observed in other related process models. The measure grounded on completed trace equivalence is sensitive to these differences as it considers all possible execution paths of a process. Instead, behavioural profile consistency enables detection and quantification of behavioural inconsistencies of an alignment, but is not affected by extensions of process models as they are observed in practise. This suggests that behavioural profile consistency is particularly suited to quantify behaviour consistency of aligned process models. These findings are in line with the results on consistency perception presented in Section 6.3.

7.4 CONSISTENT CHANGE PROPAGATION

This section turns the focus on process model synchronisation. In Section 1.3, we motivated the analysis of behaviour consistency with, among other reasons, support of change propagation between process models. As business operations continuously undergo changes, process models capturing these operations tend to drift apart.

In this section, we present a technique that supports consistent change propagation in the line of the aforementioned measures. First, we introduce an example and explain the general idea behind our approach. Then, we elaborate on the major steps of our approach in detail.

Overview of the Change Propagation Approach

Given two process models that are aligned by correspondences, our approach addresses the question of how to support a process analyst in keeping both models in sync. For a change in a source model, we isolate a *change region* in the target model. In this way, a process analyst can quickly assess the necessity to propagate the change. If change propagation seems to be appropriate, the change region spots the position where to update the process model.

The general idea of our approach can be summarised as follows. A change operation in the source process model is reflected in its behavioural profile. Therefore, the profile relations localise the change in the source model. Under the assumption that a certain share of correspondences between both models is consistent, the profile relations for corresponding activities in the target model are exploited to localise the change region. Initially, the change region covers the whole target model. Then, the change region is narrowed using two reductions. First, the change region is reduced based on *boundary transitions* that directly precede or succeed the change in strict order. Second, reduction is based on *inter-boundary transitions* that are exclusive or in interleaving order to the change. Our approach leverages the relations of the behavioural profile and neglects the co-occurrence relation of the causal behavioural profile. To localise a certain transition, the relations on the order of potential occurrence are most important. At the end, we discuss informally what can be gained by taking co-occurrences into account.

For illustration, consider the two net systems depicted in Figure 54. Both systems have been aligned by correspondences between the highlighted transitions according to their labels. The alignment is partial, e. g., transition J of system (b) is not aligned, and comprises several complex correspondences. For instance, there is a correspondence between transitions $\{C1, C2\}$ of system (a) and transitions $\{C3, C4\}$ of system (b). The alignment shows a rather high degree of behavioural profile consistency of $\mathcal{PC} = \frac{58+92}{64+100} \approx 0.91$. Most profile relations between pairs of aligned transitions are type equivalent to those observed for the corresponding transitions. Nevertheless, the alignment is not behavioural profile consistent. We observe deviations in the two behavioural profiles that relate to transition E in both systems.

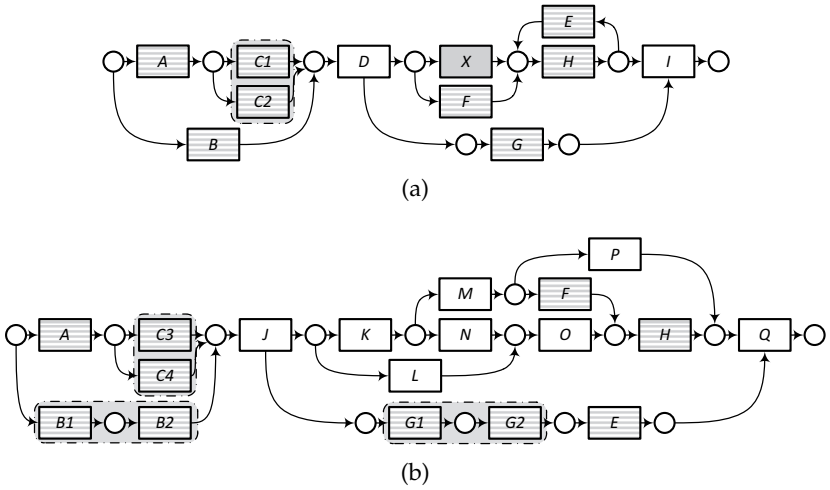


Figure 54: Two aligned net systems. Correspondences between transitions that are highlighted are indicated by the respective labels. Net system (a) has just been changed by inserting transition X.

Further, transition H is interleaving order to itself in system (a), whereas the corresponding transition in system (b) is exclusive to itself.

Assume that system (a) has just been changed by inserting transition X. Then, the question of how to spot the respective change region in system (b) has to be answered.

The Notions of a Change & a Change Region

As a first step, we discuss how a change is materialised in a net system and introduce the notion of a change region. In general, every change of the syntax of a net system may be considered for change propagation. Against the background of process models that serve different purposes, the question of what constitutes a change may be answered on a more abstract level. That is, multiple changes of the syntax of a net system are often semantically related and jointly realise a high-level change operation. Albeit defined in the context of process instance adaptation, common high-level changes have been classified as *change patterns* [488].

For our approach, we abstract from the actual implementation of a change. We assume that a change can be localised by a dedicated transition, called *change transition*. This captures not only insertion of transitions or subnets, but is also applicable for other changes. For instance, removal of a transition or a subnet may also be localised by a dedicated transition. Since we exploit the relations of the behavioural profile for change propagation, we also define a notion of a consistent change. A consistent change is defined relative to a subset of transitions, for which the rela-

tions of the behavioural profile are not affected by the change implementation. Hence, even complex change operations such as the parallelisation of a whole sequence of transitions can be addressed by our approach. Such a change just requires selecting one transition as the change transition and to identify the transitions for which the relations of the behavioural profile are not changed by the parallelisation.

Definition 7.4.1 (Change Transition, Consistent Change)
Let $S_1 = (N_1, M_1)$ be a net system with $N_1 = (P_1, T_1, F_1)$. Implementation of a change yields a system $S_2 = (N_2, M_2)$ with $N_2 = (P_2, T_2, F_2)$ and $T_1 \cap T_2 \neq \emptyset$. The transition $t \in T_2$ representing the change is referred to as the *change transition*. The change from S_1 to S_2 is *consistent* over $T' \subseteq T_1 \cap T_2$, if the behavioural profile $\mathcal{B}_1$ of S_1 over T' coincides with the behavioural profile $\mathcal{B}_2$ of S_2 over T'.

For our example in Figure 54, the aforementioned change operation involved inserting a transition and two flows. This change is localised by the change transition X. Further, the change is consistent with respect to all transitions of the net system. The relations of the behavioural profile have been not changed by the insertion of transition X.

To propagate a change from the source net system to the target net system, we identify a change region. Formally, such a change region is captured by a subnet of the target net system. We introduced the notion of a subnet in Section 2.2. For a net $N = (P, T, F)$, a place-bordered subnet may be characterised by a set of transitions $T' \subseteq T$. The subnet induced by T' is defined as $N' = (P', T', F')$ with $P' = \{p \in P \mid \exists\ t \in T' \ [\ p \in {\bullet t} \ \vee \ p \in t^{\bullet}\ ]\}$ and $F' = F \cap ((P' \times T') \cup (T' \times P'))$. Our approach builds upon change regions that are subnets induced by a set of transitions. For brevity, therefore, we refer to a set of transitions as a change region.

Extraction of a Consistent Sub-Alignment

To localise a change region in one system for a change in another system, we exploit the relations of the behavioural profile for corresponding transitions. We may only exploit correspondences for which the profile relations are consistent. Inconsistencies in these relations would lead to contradicting information on how to localise the change region.

We cope with this issue using the notion of a weak consistent sub-alignment. Given an alignment between two net systems, a weak consistent sub-alignment is a restriction of the correspondence relation and the confidence function, such that the resulting alignment is weak behavioural profile consistent. We rely

on weak behavioural profile consistency as this criterion is sufficient for consistent change propagation. Our approach does not leverage self-relations, so that those are not required to be consistent.

Definition 7.4.2 (Weak Consistent Sub-Alignment)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, and $(\sim, \zeta)$ an alignment with $\sim \subseteq T_1 \times T_2$. The tuple $(\sim', \zeta')$ is a *weak consistent sub-alignment* of $(\sim, \zeta)$, iff $\sim' \subset \sim$, $\forall\, (t_x, t_y) \in \sim' \;\; [\; \zeta'(t_x, t_y) = \zeta(t_x, t_y)\;]$, and $\sim'$ is weak behavioural profile consistent.

An inconsistent alignment may be restricted in different ways to yield a weak consistent sub-alignment. It seems reasonable to rely on sub-alignments that comprise a maximal number of elementary correspondences. However, there may be multiple sub-alignments that show the maximal number of elementary correspondences. Further, identification of maximal alignments is an optimisation problem, which is computationally hard. Search algorithms, such as the A^*-algorithm [108], may be used to find maximal sub-alignments. Besides the number of elementary correspondences, the sum of the confidence values related to the correspondences may guide the selection of a sub-alignment. Again, this can be seen as an optimisation problem. As a non-optimal sub-alignment only lowers the amount of information exploited to support change propagation, we do not consider this to be a severe problem. Hence, non-optimal heuristics may be applied to select a weak consistent sub-alignment.

As discussed earlier, we assume a change represented by a change transition t_x to be consistent over a certain set of transitions T' of the source net system. All elementary correspondences $(t_1, t_2) \in \sim$ that relate to inconsistent transitions, $t_1 \notin T'$, are removed from the alignment before a weak consistent sub-alignment is selected. Hence, the set of correspondences that may be exploited for change propagation is reduced twice – once based on inconsistencies in the source system caused by the change implementation and once based on inconsistencies between the aligned net systems.

The change applied to system (a) in Figure 54, i. e., insertion of transition X, is consistent. The alignment between both systems is not weak behavioural profile consistent, though. We introduce formal means to identify a root cause of inconsistency when focussing on behaviour consistency between a net system and observed execution sequences. Now, we rely on manual investigation of profile relations that are not type equivalent in both systems. This reveals that the inconsistencies are mainly caused by transition E in either system. Hence, we remove the respective correspondence from the alignment. Actually, the ob-

tained sub-alignment is weak behavioural profile consistent and can be leveraged for change propagation.

Reduction based on Boundary Transitions

Initially, we consider all transitions of the target system to form the change region, i. e., to induce a place-bordered subnet in which the according change shall be realised. The change region is narrowed by reducing this set of transitions. The first reduction requires the identification of the closest aligned transitions that are in strict order with the change transition in the source system. We refer to these transitions as *preceding* or *succeeding boundary transitions*. Here, closest means that there must not be any aligned transitions between a boundary transition and the change transition, which is also in strict order with the latter. We capture boundary transitions, either of the source system or of the target system, as follows.

Definition 7.4.3 (Boundary Transitions)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2, \}$ their behavioural profiles, and $\sim \subseteq T_1 \times T_2$ a weak behavioural profile consistent correspondence relation. Let t_x be a change transition of a consistent change in S_1.

- The set of *preceding boundary transitions* $PBT_1 \subseteq T_1^{\sim}$ of S_1 contains all aligned transitions that directly precede t_x in strict order, $PBT_1 = \{t_1 \in T_1^{\sim} \mid t_1 \rightsquigarrow_1 t_x \wedge \forall\, t_2 \in T_1^{\sim} [\, t_2 \rightsquigarrow_1 t_x \Rightarrow t_1 \not\rightsquigarrow_1 t_2 \,]\}$. The set of *succeeding boundary transitions* SBT_1 of S_1 is defined accordingly.
- The set of *preceding boundary transitions* $PBT_2 \subseteq T_2^{\sim}$ of S_2 contains all corresponding transitions for preceding boundary transitions that are not succeeded in strict order by another corresponding transition, $PBT_2 = \{t_1 \in T_2^{\sim} \mid \exists\, t_2 \in PB_1 [\, t_2 \sim t_1 \wedge \forall\, t_3 \in T_2^{\sim} [\, t_2 \sim t_3 \Rightarrow t_1 \not\rightsquigarrow_2 t_3 \,]]\}$. The set of *succeeding boundary transitions* SBT_2 of S_2 is defined accordingly.

We illustrate the concept of boundary transitions using our running example depicted in Figure 54. System (a) shows three preceding boundary transitions, namely B, C1, and C2. All of them are in strict order with the change transition X. Transition E does not qualify to be a preceding boundary transition, as we removed the respective correspondence from the alignment to ensure weak behavioural profile consistency. Transition H is the only succeeding boundary transition in system (a).

Using boundary transitions, the change region in the target system is narrowed. The idea behind is that strict order between boundary transitions and the change transition in the source system has to be preserved for the boundary transitions and the

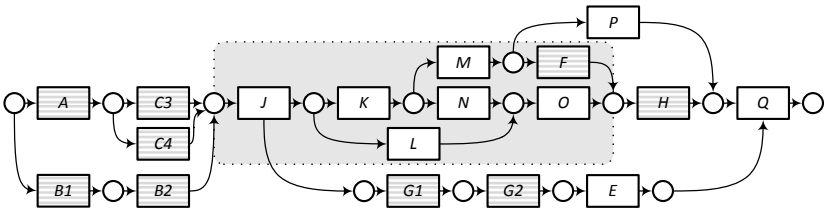


Figure 55: The change region obtained by boundary transition reduction for the setting illustrated in Figure 54.

change region in the target system. A transition is removed from the change region, if it meets one of the following requirements.

- It precedes a preceding boundary transition in strict order.
- It succeeds a succeeding boundary transition in reverse strict order.
- It is exclusive to or in interleaving order with a boundary transition.

Formally, we define this reduction of the change region as follows.

Definition 7.4.4 (Boundary Transition Reduction)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2, \}$ their behavioural profiles, and $\sim \subseteq T_1 \times T_2$ a correspondence relation. Let t_x be a change transition in S_1, PBT_2 and SBT_2 the sets of preceding and succeeding boundary transitions of S_2, and $B_2 = PBT_2 \cup SBT_2$. The *boundary transition reduction* creates a change region $\mathcal{T}_C$, such that

$$
\begin{aligned}
&\mathcal{T}_C = T_2 \setminus PT \setminus ST \setminus BT \quad \text{with} \\
&PT = \{t_1 \in T_2 \mid \exists\, t_2 \in PBT_2 \,[\, t_1 \rightsquigarrow_2 t_2 \,]\} \\
&ST = \{t_1 \in T_2 \mid \exists\, t_2 \in SBT_2 \,[\, t_2 \rightsquigarrow_2 t_1 \,]\} \\
&BT = \{t_1 \in T_2 \mid \exists\, t_2 \in B_2 \,[\, (t_1 +_2 t_2) \;\vee\; (t_1 \|_2 t_2) \,]\}
\end{aligned}
$$

We illustrate the boundary transition reduction with our running example, introduced in Figure 54. Figure 55 depicts the target system and highlights the change region once boundary transition reduction has been applied to the set of all transitions of the net system. For instance, transitions A and Q are removed as they precede a preceding boundary transition, or succeed a succeeding boundary transition, respectively. Transition P is removed from the change region, as it is exclusive to the boundary transition H. Further, all boundary transitions are not part of the obtained region either. Technically, they are removed by the reduction since their self-relation is either exclusiveness or interleaving order.

Reduction based on Inter-Boundary Transitions

The second kind of reduction of the initial change region exploits transitions that are in strict order with preceding and succeeding boundary transitions in the source system, but not with the change transition. We refer to these transitions as *inter-boundary transitions*. Those transitions are between the boundary transitions in terms of strict order and show exclusiveness or interleaving order with respect to the change transition. To keep the formalisation concise, we capture inter-boundary transitions only for the target system.

Definition 7.4.5 (Inter-Boundary Transitions)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2, \}$ their behavioural profiles, and $\sim \subseteq T_1 \times T_2$ a weak behavioural profile consistent correspondence relation. Let t_x be a change transition of a consistent change in S_1 and PBT_2 and SBT_2 the sets of preceding and succeeding boundary transitions of S_2.

- The set of *exclusive inter-boundary transitions* $EIT \subseteq T_2^{\sim}$ of S_2 contains all aligned transitions that precede and succeed the boundary transitions and for which the corresponding transitions are exclusive to the change transition, $EIT = \{t_1 \in T_2^{\sim} \mid \forall\, t_2 \in T_1^{\sim} \,[\, t_2 \sim t_1 \Rightarrow t_2 +_1 t_x \,] \wedge \forall\, t_p \in PBT_2, t_s \in SBT_2 \,[\, t_p \rightsquigarrow_2 t_1 \rightsquigarrow_2 t_s \,]\}$.
- The set of *interleaving inter-boundary transitions* $IIT \subseteq T_2^{\sim}$ of S_2 contains all aligned transitions that precede and succeed the boundary transitions and for which the corresponding transitions are in interleaving order with the change transition, $EIT = \{t_1 \in T_2^{\sim} \mid \forall\, t_2 \in T_1^{\sim} \,[\, t_2 \sim t_1 \Rightarrow t_2 \|_1 t_x \,] \wedge \forall\, t_p \in PBT_2, t_s \in SBT_2 \,[\, t_p \rightsquigarrow_2 t_1 \rightsquigarrow_2 t_s \,]\}$.

For our example, see Figure 54, we identify one inter-boundary transition in system (b), i. e., transition F. It is located between the boundary transitions with respect to strict order. Further, the corresponding transition in system (a) is exclusive to the change transition. Transitions G1 and G2 do not qualify as inter-boundary transitions. Even though there is a strict order dependency from all preceding boundary transitions to {G1, G2}, both transitions are in interleaving order with the succeeding boundary transition H.

As for boundary transition reduction, inter-boundary transitions are leveraged to narrow the change region in the target system. We remove a transition from the change region, if it meets one of the following requirements.

- It is not exclusive to one of the exclusive inter-boundary transitions.

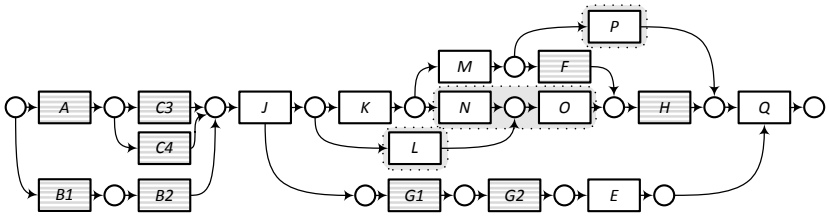


Figure 56: The change region obtained by inter-boundary transition reduction for the setting illustrated in Figure 54.

- It is not in interleaving order with one of the interleaving inter-boundary transitions.
- It is an exclusive or interleaving inter-boundary transition.

We define the inter-boundary transition reduction as follows.

Definition 7.4.6 (Inter-Boundary Transition Reduction)
Let $S_1 = (N_1, M_1)$ and $S_2 = (N_2, M_2)$ be net systems with $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$, $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2, \}$ their behavioural profiles, and $\sim \subseteq T_1 \times T_2$ a correspondence relation. Let t_x be a change transition of a consistent change in S_1 and EIT and IIT the sets of exclusive and interleaving inter-boundary transitions of S_2. The *inter-boundary transition reduction* creates a change region $\mathcal{T}_C$, such that

$$\mathcal{T}_C = T_2 \setminus EIT \setminus IIT \setminus ET \setminus IT \quad \text{with}$$
$$ET = \{t_1 \in T_2 \mid \exists\, t_2 \in EIT \,[\, (t_1, t_2) \notin +_2 \,]\}$$
$$IT = \{t_1 \in T_2 \mid \exists\, t_2 \in IIT \,[\, (t_1, t_2) \notin \|_2 \,]\}$$

Applying the inter-boundary reduction to our examples yields the change region highlighted in the target system in Figure 56. Four transitions, $\{L, N, O, P\}$, are exclusive to the inter-boundary transition F. The latter is not part of the change region. Hence, the change region is the place-bordered subnet induced by the transitions $\{L, N, O, P\}$.

Derivation of the Change Region

After we discussed the elementary steps of our approach, the complete algorithm to derive the change region is shown in Algorithm 4. Both reductions narrow the change region induced by a set of transitions in the target system. Thus, the final change region is the intersection of their results. Although both reductions are independent of each other, the result of one reduction may be used as the input for the second reduction. Both reductions, however, are not redundant. This has been illustrated by our running example. The change regions highlighted in Figure 55 and Figure 56 overlap only partially.

The result of our algorithm to derive the change region may either be a set of transitions of the target system or an empty

Algorithm 4: Derivation of the change region for a given change in one of two aligned net systems.

Input: $S_1 = (N_1, M_1)$, $S_2 = (N_2, M_2)$, net systems with $N_1 = (P_1, T_1, F_1)$, $N_2 = (P_2, T_2, F_2)$, and $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$, $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2, \}$ as their behavioural profiles.
$\sim \subseteq T_1 \times T_2$, a correspondence relation.
$t_x \in T_1$, a change transition.

Output: $\mathcal{T}_C$, the change region.

```
// Determine boundary and inter-boundary transitions
```
1 $PBT_1 \longleftarrow$ detPrecedingBoundTransSrc$(S_1, \mathcal{B}_1, \sim, t_x)$;
2 $SBT_1 \longleftarrow$ detSucceedingBoundTransSrc$(S_1, \mathcal{B}_1, \sim, t_x)$;
3 $PBT_2 \longleftarrow$ detPrecedingBoundTransTar$(PBT_1, S_2, \mathcal{B}_2, \sim)$;
4 $SBT_2 \longleftarrow$ detSucceedingBoundTransTar$(SBT_1, S_2, \mathcal{B}_2, \sim)$;
5 $EIT \longleftarrow$ detExclIntBoundTrans$(PBT_2, SBT_2, S_1, S_2, \mathcal{B}_1, \mathcal{B}_2, \sim)$;
6 $IIT \longleftarrow$ detInterlIntBoundTrans$(PBT_2, SBT_2, S_1, S_2, \mathcal{B}_1, \mathcal{B}_2, \sim)$;

```
// Narrow the change region by reductions
```
7 $\mathcal{T}_1 \longleftarrow$ doBoundaryTransitionReduction$(S_2, \mathcal{B}_2, PBT_2, SBT_2)$;
8 $\mathcal{T}_2 \longleftarrow$ doInterBoundaryTransitionReduction$(S_2, \mathcal{B}_2, EIT, IIT)$;

```
// Derive the change region
```
9 $\mathcal{T}_C \longleftarrow \mathcal{T}_1 \cap \mathcal{T}_2$;

set. The former indicates that there are already transitions in the system that meet the behavioural requirements with respect to the potential change. These transitions do not necessarily induce a connected subnet of the target system. Instead, there may be multiple areas in the target system that qualify for the realisation of the change. In case the set is empty, the target system does not yet contain a transition satisfying the behavioural requirements. In this case, the boundary transitions and inter-boundary transitions guide the adaptation of the target system. Boundary transitions impose requirements regarding the strict order relation. Inter-boundary transitions guide the adaptation based on the exclusiveness and interleaving order relations.

For implementing the change in the target system, the self-relation of the change transition in the source system may be exploited. Whether the change transition is exclusive to itself or in interleaving order to itself provides additional information on how to adapt the target system. Interleaving order hints at potential multiple occurrences of the transition related to the change in the source system. Such behaviour may be mirrored in the target system.

Our approach of localising a change or a change region builds upon the relations of the behavioural profile, as those specify order dependencies between transitions. Nevertheless, the presented approach can be lifted to causal behavioural profiles. Then, the change would be localised by order and co-occurrence de-

pendencies. Both types of dependencies would be required to be mirrored in the target system. However, this would require a stricter notion of consistency for the alignment. The latter would have to satisfy causal behavioural profile consistency, instead of weak behavioural profile consistency. Against the results obtained in our empirical investigation presented in Section 6.3 and the experimental evaluation done in Section 7.3, it seems questionable whether this would be beneficial in the general case. Extraction of a sub-alignment that satisfies causal behavioural profile consistency can be assumed to exclude more correspondences compared to the presented approach. Hence, change propagation would have to rely on fewer correspondences and, therefore, be less accurate. Still, co-occurrences may guide the realisation of a change in the target model even if they are not used to narrow the change region. Co-occurrences with boundary transitions in the source system hint at how to implement the change in the target system, similar to the aforementioned self-relations.

7.5 RELATED WORK

The concepts presented in this chapter mainly relate to three streams of research. In the remainder of this section, we discuss generic process model measures, measures for behavioural similarity, and techniques for process model change management.

Process Model Measures

Process model measures aim at capturing structural or behavioural characteristics of a process model. Based on these characteristics, conclusions on their understandability, maintainability, or correctness are drawn. Albeit often referred to as process model metrics in the literature, most measures are no metrics in the mathematical sense. They do not show the characteristic properties of a metric. Work on process model measures is inspired by work on measures for software complexity, see [80, 303, 527, 207, 225]. For decades, research has been conducted on complexity measures for software programs, starting with simple measures such as the number of lines of code, to measures that exploit the control flow graph and the coupling of software modules.

Surveys on how these measures have been adapted for process models can be found in [82, 263, 312]. Elementary structural measures include the number of activities (or places and transitions for a net system, respectively) and the number of flows [265]. Other structural measures consider, for instance, the number of circuits, distinct paths, and hierarchy levels, or

the maximum of the minimal distances between two activities in a process model [265, 332, 328, 262]. Behavioural measures may be based on the size of the state space of a net system or the average number of tokens over all reachable markings [265].

Besides these elementary measures, the cyclomatic number of a control flow graph has been adapted. It refers to the number of linearly independent paths in the graph [303]. Using this idea, the control-flow complexity (CFC) measure is based on the splits of a process model [81], i. e., the elements that implement the control flow routing. In contrast to the cyclomatic number, the CFC measure takes different semantics of splits into account. Further, the concepts of coupling and cohesion have been advocated as a grounding of process model measures [98, 378]. Cohesion measures may be based on the overlap of activities in terms of their input and output dependencies [378]. Coupling of activities of a process model was interpreted as the density of the process model graph, i. e., the ratio of connected activities to flows in the process model [378].

Many of the aforementioned measures have been evaluated empirically towards their correlation with the understandability, maintainability, or correctness of process models. An overview of evaluations is given in [312].

Process model measures focus on a single process model and do not aim at the comparison of two models or an alignment between them. Nevertheless, we see a close relation between these measures and our consistency measures for two reasons. First, process model measures may support analysis of behaviour consistency. Given two aligned process models, a certain measure may be computed for both process models. The delta between the obtained values indicates how much both models deviate in their general structure or behaviour. This analysis cannot reveal behavioural contradictions that we aim to detect with our measures. Instead, these measures quantify to which extent a certain level of behaviour consistency correlates with similar model structure or behaviour. This kind of analysis adds an orthogonal dimension. Even if an alignment between two process models is behaviour consistent, there may be differences in how the respective behaviour is implemented. Second, we see potential for building process model measures based on behavioural profiles and relate them to understandability, maintainability, or correctness of process models as well. For instance, a large amount of activity pairs in strict order in the behavioural profile hints at a rather simple control flow structure. The latter can be assumed to correlate with the understandability and maintainability of a process model.

Behavioural Similarity Measures

We discussed measures for behavioural similarity in Section 3.2 when reviewing techniques for matching process models to construct an alignment. In Section 7.2, we elaborated further on the conceptual differences between consistency measures and similarity measures. The former quantify the quality of the correspondence relation of an alignment and are independent of the shares of transitions that are aligned. In this section, we focus on the concrete operationalisation of behavioural similarity measures.

Various similarity measures rely on the well-known Jaccard coefficient. Given two sets A, B this coefficient defines their similarity as $sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$. Earlier, we mentioned that similarity may be assessed based on this ratio defined for (completed) traces of two net systems [107, 136]. The degree of trace consistency used as a baseline in our experimental evaluation follows this idea. It just adds projection of transitions that are not aligned to be applicable for partial alignments.

Similarities in the line of the Jaccard coefficient may be based on behavioural relations. Actually, our consistency measures incorporate this idea. We quantify behaviour consistency by the ratio of type equivalent behavioural relations to all relations between aligned transitions. Once the behavioural relations are mutually exclusive and partition a certain set of transitions, similarity quantification may be seen as a matrix comparison. Given two behavioural matrices, the ratio of consistent entries in both matrices to the size of the matrices quantifies similarity. To this end, various different matrix-based representations of behaviour may be used, e. g., behavioural profiles, footprints comprising the relations of the α-algorithm [457, 446], or causal matrices that capture transitions that precede or follow each other directly [459, 106] or indirectly [395].

Besides a ratio of traces or behavioural relations, edit distances have been used as a formal grounding for behavioural similarity measures. Following the idea of the string edit distance [267], behavioural similarity may exploit an edit distance for the language of a process model, an automata encoding this language, or an n-gram representation of the language [525, 522, 524, 523]. This idea has also been lifted to the level of change operations between process models. The authors of [271] measure similarity based on the smallest number of high-level change operations, built from change primitives, needed to transform one model into another. Although the operations are structural, they are guided by a matrix-based representation of the behaviour, an order matrix. In Section 4.5, we argued that behavioural profiles can be seen as a generalisation of the order matrix. Hence, the

similarity measures proposed in [271] may be ported to behavioural profiles.

Once the behaviour of process models is described by a state space, i. e., a labelled transition system, behavioural similarity can be measured grounded on an optimal matching between them [427, 330]. In Section 7.1, we discussed these approaches in the light of our motivating example. Both similarities assume the existence of an elementary similarity measure for state transitions. Then, similarity is determined for paths in the state space by considering the local similarity of state transitions and those of neighbouring state transitions. Due to their grounding in state spaces, the operationalisation of these similarities is not directly transferable to behavioural profiles.

Another way to operationalise a measure for behaviour similarity has been proposed for causal footprints [470, 473]. Causal footprints capture semantics for a set of transitions by two relations, look-back links and look-ahead links, see also Section 4.5. A vector space may be leveraged to assess the similarity of two process models based on their causal footprints. The space is spanned by taking all transitions, look-back links, and look-ahead links as dimensions. Models are represented as vectors according to their contained activities, and satisfied look-back and look-ahead links. In principle, the idea of using a vector-space for quantifying similarity can be transferred to any other relational semantics, such as the behavioural profile. Then, the dimensions of the vector space are derived not from look-back and look-ahead links, but from entries of the profile relations.

Process Model Change Management

Change management for process models comprises three major steps, i. e., (1) detection of differences, (2) analysis of their relations, and (3) resolution of differences [171].

Behavioural similarity measures as discussed earlier detect behavioural differences. Besides the detection that there is a behavioural deviation, several approaches try classify the deviation in terms of high-level or compound change operations [256, 124, 271, 171]. For instance, the approach presented in [256, 171] identifies compound changes that are grounded on the triconnected parse tree of a process model, the RPST [481, 359]. We discussed the RPST for WF-nets in Section 5.2. The insertion of a single-entry single-exit fragment would be an example for such a change operation. Similar change operations are identified in [271]. In [124], compound changes are identified by leveraging trace semantics of process models. Trace patterns are linked to an existing classification of process model differences [123].

Once compound changes are identified, their relation may be investigated. The detection of order or conflict dependencies between different compound changes enables fine-granular control of the change propagation [257]. Finally, the identified changes are implemented to resolve the according process model differences.

There are various conceptual differences between the techniques for process model change management and the concepts introduced in this chapter. As discussed for behavioural similarities, change management considers process models as a whole, whereas we focus on the aligned part only. Further, change management relies on well-defined change operations. In contrast, we do not make any assumption on the structure of correspondences. They may be n:m complex or overlapping. Nevertheless, there are also various commonalities with respect to the essential steps of change management. Our consistency measures (1) detect behavioural differences. Although those materialise in different behavioural relations of pairs of corresponding transitions of two aligned net systems, we do not aim at deriving compound change operations. How this could be done in principle was shown for the order matrix in [271]. Regarding step (2), our notion of a consistent sub-alignment needed for change propagation conceptually resembles the analysis of conflicting changes. An inconsistent alignment could stem from conflicting changes. Finally, step (3), the resolution of process model differences, is supported by our approach to change propagation.

Similar techniques for the detection of changes, the analysis of their relations, and their resolution have been presented in the context of process instance migration [376, 383, 488, 97]. Several of the aforementioned techniques have their roots in process evolution and adaptive process management.

In summary, our approach imposes fewer restrictions on the relation between two process models than techniques for change management. On the down side, the presented techniques do not yield a concrete change operation, but are limited to guiding the implementation of a change by identifying a change region.

7.6 CONCLUSION

This chapter introduced consistency measures for aligned net systems. Those leverage behavioural profiles and have been defined in the spirit of the Boolean consistency criteria introduced in the previous chapter. We evaluated these measures experimentally using process models of the SAP reference model. The results suggest that the measure based on (non-causal) behavioural profiles is particularly suited in this context. It provides a fine-granular measure, but is insensitive to extensions of process

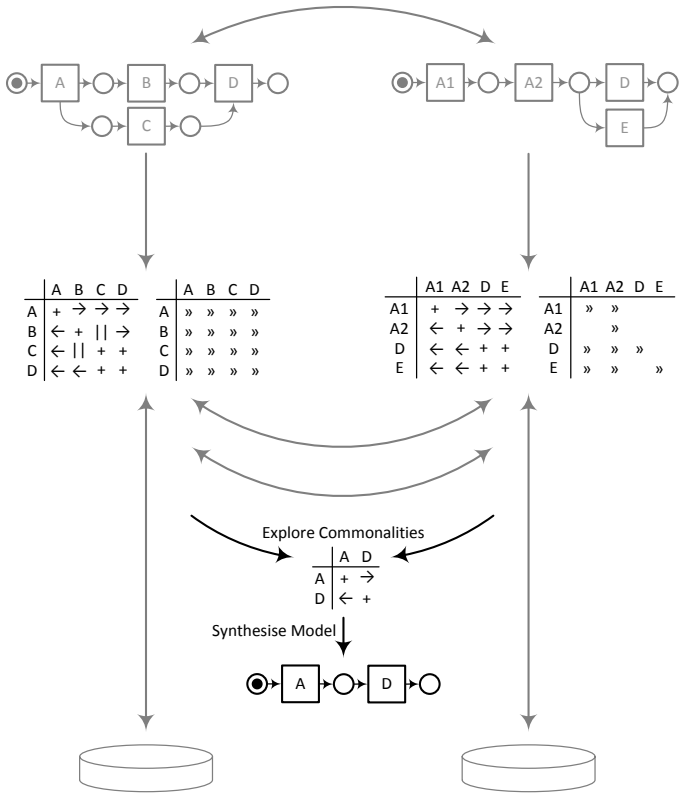
models as they are observed in practise. Finally, we focussed on consistent change propagation between process models. Given a change in one of two aligned models, we presented a technique to identify a change region in the other model. This supports a process analyst in assessing the necessity to propagate the change and, if needed, implemented it.

We elaborated on the relation of measuring behaviour consistency of an alignment and judging on behavioural similarity in detail. The conceptual difference is the focus on aligned activities of consistency measures instead of the complete models considered by similarities. Nevertheless, this difference is only a question of the operationalisation of a measure. In principle, techniques to address both questions may have similar roots and rely on the very same idea to quantify behavioural deviation. When reviewing related work, we outlined these commonalities. We foresee that there is a large potential to adapt techniques introduced for the field of consistency measurement for similarity measurement, and vice versa. As a first step in this direction, we recently proposed behavioural similarity metrics that are grounded on behavioural profiles [254, 253]. Those are inspired by the measures proposed in this chapter. Still, they focus on the complete process models instead of only the aligned activities. Further, these measures are proper metrics and, therefore, enabled the usage of efficient indexing techniques if applied for process model search [253].

8

EXPLORING PROCESS MODEL COMMONALITIES

This chapter is based on results published in [506, 423, 425, 494].



Deciding and quantifying consistency are essential tasks of behavioural analysis. In the previous chapters, we presented a spectrum of consistency notions and measures, so that consistency analysis can be gradually tailored towards a concrete setting. Once a consistency result is obtained, exploration of behavioural commonalities and differences that caused the result provides an added value. It allows for interpreting and explaining the obtained consistency values. This chapter introduces concepts and techniques that support such an explorative behavioural analysis. We start by discussing the need to perform this kind of analysis in Section 8.1. Many analysis questions can be answered using algebraic operations and relations. Those have only partially been defined based on common behaviour equivalences. To enable explorative behavioural analysis, Section 8.2 defines a set algebra for behavioural profiles. It allows

for computing with the abstracted behaviour of process models. Section 8.3 complements these results by a model synthesis for behavioural profiles. Then, Section 8.4 combines algebraic concepts and the model synthesis to address the analysis questions raised in Section 8.1. In Section 8.5, we evaluate the set algebra with an experimental setup. The experiment uses the SAP reference model known from the previous chapter. Section 8.6 reviews related work. We conclude this chapter in Section 8.7.

8.1 EXPLORATIVE BEHAVIOURAL ANALYSIS

Once consistency is decided and quantified, exploration of behavioural commonalities and differences helps to interpret the consistency results. An explorative behavioural analysis targets the question whether the encountered differences are within acceptable boundaries or whether harmonisation of the models is required. In this section, we illustrate the need for this kind of analysis by an example setting and concrete analysis questions. Then, we discuss different ways to approach these questions by means of algebraic concepts. Those may be grounded on behaviour equivalences or on behavioural relations.

An Example Setting

We take up the lead-to-order process used for illustration in the previous chapters. Figure 57 presents two more net systems that depict the process and have been aligned by correspondences. Again, both systems describe a similar processing of a lead. Nevertheless, we also observe behavioural differences. In particular, a quote may be updated, forwarded to other employees, and approved multiple times in system (a), which is not possible in system (b). These differences may stem from different modelling purposes that led to the creation of the respective models. System (a) focusses on the major steps of processing from an organisational perspective and may be used for process documentation. System (b) depicts the steps that may be implemented in an information system to support the lead-to-order process.

The behavioural differences between both systems are detected by the presented consistency measures. For the alignment shown in Figure 57, we obtain a value of behavioural profile consistency of $\mathcal{PC} = \frac{28+50}{36+64} = 0.78$. Given such a consistency value, a more detailed analysis is often needed to decide on the severity of the detected behavioural deviations. This holds in particular, once not only a pair of aligned process models, but a set of aligned models is investigated.

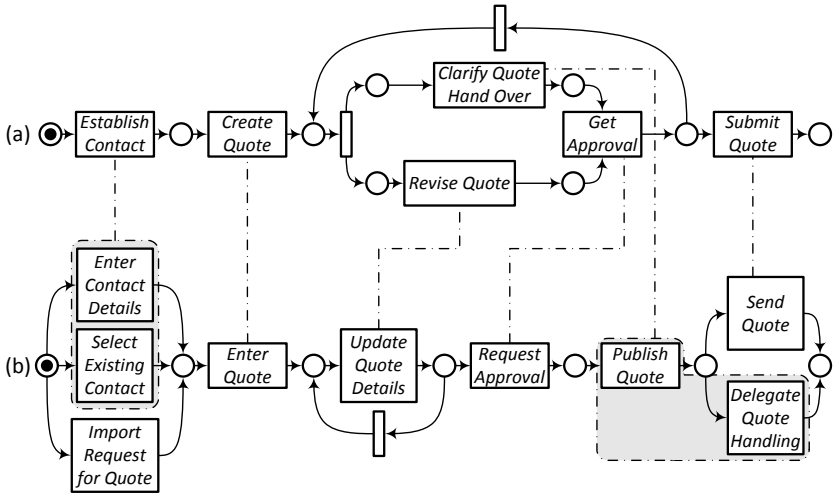


Figure 57: Two net systems depicting a lead-to-order process that are aligned by correspondences.

Analysis Challenges

To support the interpretation of behavioural deviations for a set of aligned process models, various challenges have to be addressed. We focus on four challenges, or analysis questions, respectively. Even though other aspects may be relevant as well, we consider the following questions to be common examples for this kind of analysis.

C1: Given a set of aligned models, what are their commonalities in terms of shared behaviour? This question relates to the challenge of identifying the behaviour that is invariably agreed on by all related process models, i. e., the implemented invariant behaviour.

C2: Given a set of aligned models, what is the most general allowed behaviour? The challenge to answer this question is to integrate the behaviour defined by all process models, such that the most general behaviour becomes visible. This resembles the notion of a configurable process model, see [392, 181, 259], which subsumes the behaviour defined in separate process models.

C3: Given a set of aligned models, what are their commonalities in terms of shared forbidden behaviour? Similar to the first question, also the shared forbidden behaviour is of interest to interpret consistency results obtained for aligned process models.

C4: Given a set of aligned models, which process models are redundant in terms of the specified behaviour? The challenge behind this question is to identify models for which the specified behaviour is subsumed by another process model. In that case, the existence of the former model may be challenged.

Answering these questions first requires us to choose a behavioural model to capture commonalities and differences. Then, the challenge is to come to an appropriate specification of formal

operations to support reasoning with the behaviour defined by the process models. We argue that the essential questions of an explorative behavioural analysis may be answered using set algebraic operations, such as complementation, intersection, and union, and set algebraic relations, such as equivalence and inclusion. Hence, the question is how to define a behavioural algebra for a certain behavioural model. As for consistency notions and consistency measures, two fundamentally different approaches may be taken to define algebraic concepts to compute with behaviour. Those may be grounded on behaviour equivalences or behavioural relations.

Algebraic Concepts based on Behaviour Equivalences

In the previous chapters, we reviewed work on the application of behaviour equivalences in the presence of extensions and refinements. These works can be related to particular subsets of a behavioural algebra. The notions of behaviour inheritance [138, 451, 33, 412] provide an answer to the question of inclusion of behaviour. A process model includes the behaviour of another process model if both are behaviour equivalent once extensions are blocked or hidden, cf., the discussion in Section 6.1.

Further, there has been significant work on the integration of behaviour in the sense of a union or merge operation. In Section 6.4, we discussed process views as a means to accommodate for the multitude of potential perspectives on a business process. Although process views are commonly derived from a holistic core model, the latter may also be derived by integrating multiple views [366, 343, 314]. Merge operations have also been proposed to consolidate collections of process models by configurable reference process models [330, 180, 379, 258, 259] or to resolve version conflicts of process models [255]. The concrete operationalisation of a merge operation typically depends on whether the models are assumed to represent the same business process [366, 180]. If so, they provide views that complement each other and need to be synchronised, see [366, 343, 314]. Such an operationalisation is close to work on merging partial behavioural models, aka scenarios, used in Software Engineering [443]. In contrast to view integration, however, scenarios may not only capture complementary, but also alternative behaviour of a process. If the models are not assumed to represent the same business process, the merged model implements choices between the varying behaviour observed in different models. This kind of operationalisation is followed in [366, 343, 330, 180, 379, 258, 259]. Recently, the extraction of a digest of the merged process model has been proposed [259]. Given the merged model, this approach extracts the most recurrent elements of the original pro-

cess models. Hence, this digest can be seen as a notion of intersection.

Despite these partial results, we miss an overarching behavioural algebra that incorporates the mentioned concepts. To the best of our knowledge, the only attempt to define algebraic concepts was reported in [343]. The authors introduce selection, projection, join, union, and difference operators for net systems. Still, the concepts are defined on the net system's syntax, not its semantics.

Under the assumption of finite sets of traces, one may imagine to define a behavioural algebra using standard set algebraic operations and relations applied to sets of traces. Nevertheless, the question of how to cope with complex correspondences and with activities that are not aligned must be addressed. In Section 6.1, we discussed that this is far from trivial.

Algebraic Concepts based on Behavioural Relations

Behavioural relations are sets of ordered pairs of activities. Thus, we may apply standard set algebraic operations and relations directly to behavioural relations. However, this does not answer the question of how to define algebraic concepts for a *set* of behavioural relations. We are not aware of any work that proposes an algebra over behavioural relations of process models.

For the definition of algebraic concepts, again, different behavioural semantics may be leveraged. We proposed consistency notions and measures based on behavioural profiles. We motivated our choice for indirect dependencies by the observation that those are insensitive to extensions. For the same reason, it seems appropriate to define algebraic concepts based on behavioural profiles to address the challenges of explorative behavioural analysis in our context.

8.2 A SET ALGEBRA FOR BEHAVIOURAL PROFILES

This section is dedicated to a formal definition of a set algebra for behavioural profiles. Our algebra is based on non-causal behavioural profiles and neglects co-occurrences of transitions. This is motivated by the results obtained in Section 6.3 and Section 7.3. We observed that causal dependencies between activities may be broken once only the *happy path* of business operations is captured, which suggests to neglect co-occurrences when evaluating the quality of an alignment. Hence, exploring behavioural commonalities and differences shall focus on the order of potential occurrence of activities, whereas causal dependencies are of secondary importance.

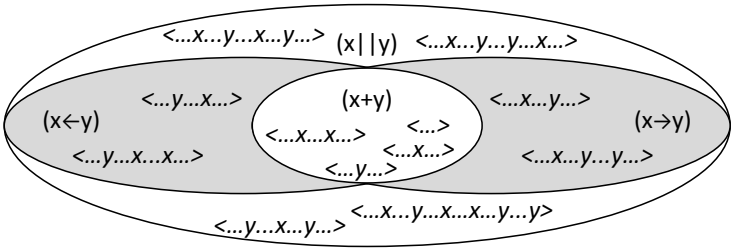


Figure 58: Sets of traces induced by the relations of the behavioural profile for two transitions x and y.

Apart from that, the definition of algebraic concepts is challenging only for the relations of the behavioural profile. The profile relations are interrelated. This aspect has to be incorporated into the definition of algebraic concepts. In contrast, co-occurrences are captured in dichotomous way that is largely independent of the profile relations. Therefore, standard set algebraic concepts can be applied to cope with the co-occurrence relation of the causal behavioural profile.

To compute with behaviour of aligned net systems, we first discuss a notion of strictness of the relations of the behavioural profile. This strictness is the foundation for the definition of the set algebra. Second, we discuss a normalisation of behavioural profiles. It abstracts from complex correspondences between aligned net systems by identifying a dominating behavioural relation for sets of transitions. Then, we introduce set-theoretic relations and set-theoretic operations. Finally, we investigate properties of the introduced operations.

Strictness of Behavioural Relations

The relations of the behavioural profile can be classified according to their *strictness*. They allow different levels of freedom for the occurrences of transitions in a firing sequence starting in the initial marking of a net system. Interleaving order can be seen as the absence of any restriction on the order of potential occurrence – the transitions are allowed to appear in an arbitrary order. In contrast, (reverse) strict order defines a particular order of occurrence for two transitions. Exclusiveness prohibits the occurrence of two transitions together in one firing sequence. Therefore, we consider interleaving order to be the weakest relation, and exclusiveness to be the strictest relation.

For a pair of transitions (x, y), this strictness is reflected in the containment hierarchy of firing sequences that show either none, one, or both transitions, illustrated in Figure 58. Here, all firing sequences that conform to a specific relation of the behavioural profile are part of the encircled set of firing sequences. A net

system that defines interleaving order between two transitions x and y allows for any firing sequence. A firing sequence may contain none, one, or both transitions in any order. A net system that imposes exclusiveness for both transitions is most restrictive. It allows for firing sequences that comprise none or only one of the transitions. This set is a proper subset of the firing sequences induced by interleaving order. Strict order and reverse strict order are intermediate relations.

This notion of strictness is the foundation for the definition of a set algebra for behavioural profiles. Our operations and relations are *not* defined based on sets of traces, but on the relations of the behavioural profile. Still, the strictness of behavioural relations illustrated with sets of traces motivates the way we define the set-algebraic relations and operations.

Normalisation of Behavioural Profiles

To compute with behavioural profiles of multiple aligned net systems, we require the behavioural profiles to be defined only over aligned transitions. Further, we assume that the correspondence relation between net systems is functional and injective, i. e., there are no complex correspondences. As this requirement can rarely be assumed to hold in practise, we define a normalisation of a behavioural profile. Given two sets of transitions of one system that are part of a complex correspondence defined between this system and another system, the normalisation determines their dominating behavioural relation.

The derivation of a dominating profile relation is formalised in Algorithm 5. It takes a net system, its behavioural profile, two sets transitions, and a normalisation threshold as input. The algorithm then returns the dominating profile relation for the sets of transitions.

Derivation of the dominating profile relation is based on the strictness hierarchy discussed in the previous section. First, we compute the share of transition pairs that is covered by the strictest relation, exclusiveness, or by the two strictest relations, exclusiveness and (reverse) strict order (lines 2 to 4). Then, we select the dominating profile relation according to these ratios and a normalisation threshold (lines 5 to 8). The idea is to select the strictest relation that is in line with the profile relations for the requested share of transition pairs. For instance, a threshold of $t = 0.9$ yields a dominating profile relation, such that 90% of the profile relations between transition pairs are type equivalent or stricter according to the aforementioned hierarchy.

Algorithm 5 defines how to determine a dominating profile relation for two non-equal sets of transitions. These sets may be

Algorithm 5: Derivation of the dominating profile relation for two sets of transitions.

Input: $S = (N, M)$, a net system with $N = (P, T, F)$ and $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ as its behavioural profile.
$T_x, T_y \subseteq T$, $T_x \neq T_y$, two sets of transitions.
t, normalisation threshold with $0 < t \leqslant 1$.

Output: $R \in \{\rightsquigarrow, \rightsquigarrow^{-1}, +, \|\}$, the type of the dominating profile relation for the pair (T_x, T_y).

$TP \longleftarrow (T_x \times T_y) \cup (T_y \times T_x)$;

// Determine ratios for profile relations

$r_{+} \longleftarrow \frac{|+ \cap TP|}{|TP|}$;

$r_{\rightsquigarrow} \longleftarrow \frac{|(+ \cap TP) \cup (\rightsquigarrow \cap (T_x \times T_y)) \cup (\rightsquigarrow^{-1} \cap (T_y \times T_x))|}{|TP|}$;

$r_{\rightsquigarrow^{-1}} \longleftarrow \frac{|(+ \cap TP) \cup (\rightsquigarrow^{-1} \cap (T_x \times T_y)) \cup (\rightsquigarrow \cap (T_y \times T_x))|}{|TP|}$;

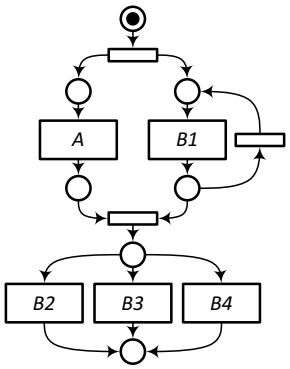
// Select profile relation according to ratios

if $r_{+} \geqslant t$ **then return** $+$;
if $r_{\rightsquigarrow} \geqslant t$ **then return** $\rightsquigarrow$;
if $r_{\rightsquigarrow^{-1}} \geqslant t$ **then return** $\rightsquigarrow^{-1}$;
return $\|$;

partially overlapping. This is the case, if the sets of transitions are induced by overlapping correspondences of an alignment.

To normalise a behavioural profile, we also have to determine a dominating self-relation for a set of transitions. A transition is either exclusive or in interleaving order to itself. For a set of transitions, we select the relation that is observed most frequently as a self-relation for the respective transitions.

Normalisation of behavioural profiles of two aligned net systems results in behavioural profiles that are defined over sets of transitions that are part of correspondences. We illustrate



(a)

	A	B1	B2	B3	B4
A	+	||	→	→	→
B1	||	||	→	→	→
B2	←	←	+	+	+
B3	←	←	+	+	+
B4	←	←	+	+	+

(b)

	A	B
A	+	→
B	←	+

(c)

Figure 59: Net system (a) and its behavioural profile (b). Normalisation of the behavioural profile for the sets of transitions $\{A\}$ and $\{B1, B2, B3, B4\}$ yields the profile illustrated in (c).

this normalisation with the example depicted in Figure 59. Figure 59a shows a net system and Figure 59b depicts its behavioural profile. Assume that the system is aligned with another net system by two correspondences, one involving only transition A and one comprising four transitions {B1, B2, B3, B4}. Then, we normalise the behavioural profile for the sets of transitions {A} and {B1, B2, B3, B4}. Taking a normalisation threshold of $t = 0.7$, the resulting profile is depicted in Figure 59c. The strict order dependency between transition A and each of the transitions {B2, B3, B4} leads to strict order as the dominating relation for the two sets of transitions. The interleaving order dependency between transitions A and B1 is neglected. For a normalisation threshold of $t = 1$, however, this relation would be considered. Then, the normalisation yields an interleaving order dependency as the dominating relation between the sets of transitions {A} and {B1, B2, B3, B4}. Further, we decide on exclusiveness as the self-relation for both sets of transitions. This stems from the observation that most of the transitions {B1, B2, B3, B4} show exclusiveness as their self-relation.

Normalisation may result in behavioural profiles that are inconsistent in the sense that they specify contradicting information. We discuss these issues in detail as part of the model synthesis from a behavioural profile.

Set-Theoretic Relations

We start the definition of our algebra by introducing three relations, i. e., equivalence, inclusion, and emptiness for behavioural profiles. We reason only on behavioural profiles that are defined over aligned transitions that are part of at least one correspondence. Transitions that are not aligned by any correspondence are neglected. Further, we assume that behavioural profiles have been normalised according to the correspondences between the aligned net systems as discussed in the previous section. This allows us to keep the formalisation concise by abstracting from the actual correspondence relation. We assume corresponding transitions to be identical.

Equivalence. Two behavioural profiles are equivalent, if they enforce equal behavioural dependencies for all transitions. The equivalence of behavioural profiles requires equivalence of the profile relations. Note that the notion of behavioural profile equivalence for net systems as defined in Definition 4.4.1 imposes the same requirements. Further, we discussed in Section 4.4 that equivalence of behavioural profiles does not imply equal trace semantics for two net systems.

Definition 8.2.1 (Equivalence of Behavioural Profiles)
Let $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2\}$ be behavioural pro-

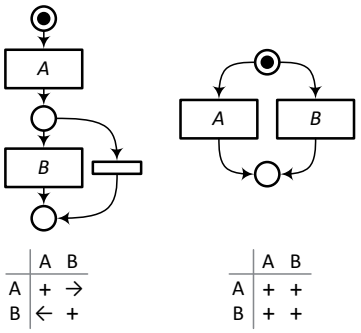


Figure 60: The behavioural profile of net system (a) includes the one of net system (b).

files over a finite set of transitions T. $\mathcal{B}_1$ *equals* $\mathcal{B}_2$, denoted by $\mathcal{B}_1 = \mathcal{B}_2$, iff their relations are equal for all pairs of transitions, $\rightsquigarrow_1 = \rightsquigarrow_2$, $+_1 = +_2$, and $\|_1 = \|_2$.

Inclusion. An inclusion holds between two behavioural profiles, if one profile subsumes the behavioural dependencies of another profile according to the notion of strictness discussed earlier.

Definition 8.2.2 (Inclusion of Behavioural Profiles)
Let $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2\}$ be behavioural profiles over a finite set of transitions T. $\mathcal{B}_1$ *includes* $\mathcal{B}_2$, denoted by $\mathcal{B}_1 \subseteq \mathcal{B}_2$, iff for all pairs of transitions $(t_x, t_y) \in T \times T$ it holds that

- $t_x +_1 t_y$ implies $t_x +_2 t_y$,
- $t_x \rightsquigarrow_1 t_y$ implies $t_x +_2 t_y$ or $t_x \rightsquigarrow_2 t_y$,
- $t_x \rightsquigarrow_1^{-1} t_y$ implies $t_x +_2 t_y$ or $t_x \rightsquigarrow_2^{-1} t_y$.

If $\mathcal{B}_1 \subseteq \mathcal{B}_2$, but not $\mathcal{B}_1 = \mathcal{B}_2$, we speak of proper inclusion, denoted by $\mathcal{B}_1 \subset \mathcal{B}_2$.

Figure 60 illustrates inclusion of behavioural profiles. The profile of the net system in Figure 60a includes the one of the net system in Figure 60b, since the former is less restrictive. Here, we neglect the unlabelled transition in Figure 60a which is without counterpart in the other system. The profile in Figure 60a allows for ordered occurrence of transitions A and B in a firing sequence starting in the initial marking. The profile in Figure 60b, in turn, forbids any occurrence of both transitions in a firing sequence. This example illustrates that inclusion of behavioural profiles does not imply inclusion of the respective sets of traces, which stems from the assumed behavioural abstraction. Apparently, the trace $\sigma = \langle B \rangle$ of the system in Figure 60b is not possible in the system in Figure 60a.

The behavioural abstraction allows us to cope with behavioural deviations as they are visible in our initial example in

Figure 57. In system (a), transition 'Clarify Quote Hand Over' may occur before 'Get Approval'. Further, both transitions may occur multiple times in a firing sequence starting in the initial marking, so that they are in interleaving order according to the behavioural profile. In contrast, the corresponding transitions in system (b) are in reverse strict order. Hence, the behavioural dependencies imposed by system (b) are included in the dependencies imposed by model (a). Most approaches that built upon a trace-based assessment will fail to address such cases.

Emptiness. A behavioural profile is empty, if it defines all pairs of transitions to be exclusive. Such a profile forbids the occurrence of any two transitions. This definition is motivated by the idea that one may add a transition to any net system that carries no meaning but indicates initialisation of the business process. Then, the exclusiveness of all transitions with this transition would imply that no transition representing a business activity is allowed to occur.

Definition 8.2.3 (Emptiness of a Behavioural Profile)
Let $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ be a behavioural profile over a finite set of transitions T. $\mathcal{B}$ is *empty*, iff all pairs of transitions are exclusive, $+ = T \times T$.

Set-Theoretic Operations

As a next step, we introduce three set operations for behavioural profiles, i. e., complementation, intersection, and union. Again, we abstract from a correspondence relation between net systems. We assume that behavioural profiles are defined over identical sets of transitions once more than one profile is considered.

Complementation. The complement operation is defined for a single behavioural profile and returns a profile that specifies reverse relations for all transition pairs of the original profile.

Definition 8.2.4 (Complement of a Behavioural Profile)
Let $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ be a behavioural profile over a finite set of transitions T. The *complement* $\overline{\mathcal{B}} = \{\rightsquigarrow', +', \|'\}$ of $\mathcal{B}$ is a behavioural profile over T, such that for all pairs of transitions $(t_x, t_y) \in T \times T$ it holds that

- $t_x +' t_y$, iff $t_x \| t_y$,
- $t_x \rightsquigarrow' t_y$, iff $t_x \rightsquigarrow^{-1} t_y$,
- $t_x \|' t_y$, iff $t_x + t_y$.

The complement operation is illustrated in Figure 61. The system in Figure 61a and the one in Figure 61b show complementary behavioural profiles. The profile of the one system is the complement of the profile of the other system, and vice versa. For instance, there is a strict order dependency between trans-

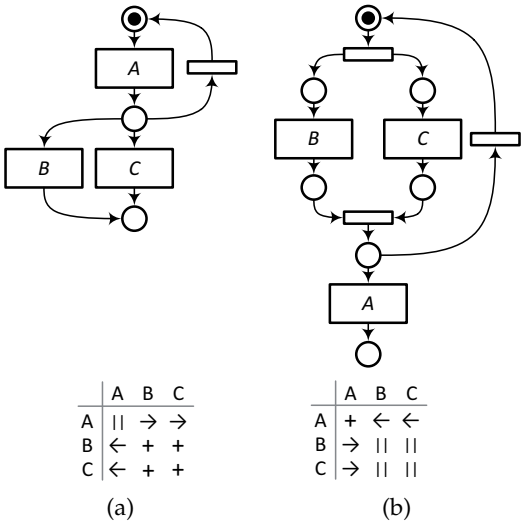


Figure 61: The behavioural profiles of both net systems are complementary.

itions A and B in the system in Figure 61a, whereas both transitions are in reverse strict order in the profile of the system in Figure 61b. Further, transition A may occur multiple times according to the profile in Figure 61a, whereas it is exclusive to itself in the profile in Figure 61b.

Intersection. Given two behavioural profiles, the intersection operation yields a third behavioural profile that combines the strictest relations of the behavioural profiles. Hence, the intersection represents the behavioural dependencies that are shared by both profiles.

Definition 8.2.5 (Intersection of Behavioural Profiles)
Let $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2\}$ be behavioural profiles over a finite set of transitions T. The *intersection* of these profiles is a behavioural profile $\mathcal{B}_3 = \{\rightsquigarrow_3, +_3, \|_3\}$ over T, denoted by $\mathcal{B}_1 \cap \mathcal{B}_2 = \mathcal{B}_3$, such that for all pairs of transitions $(t_x, t_y) \in T \times T$ it holds that

- $t_x +_3 t_y$, iff either $t_x +_1 t_y$, $t_x +_2 t_y$, $(t_x \rightsquigarrow_1 t_y \wedge t_x \rightsquigarrow_2^{-1} t_y)$, or $(t_x \rightsquigarrow_1^{-1} t_y \wedge t_x \rightsquigarrow_2 t_y)$,
- $t_x \rightsquigarrow_3 t_y$, iff either $(t_x \rightsquigarrow_1 t_y \wedge (t_x \rightsquigarrow_2 t_y \vee t_x \|_2 t_y))$ or $(t_x \rightsquigarrow_2 t_y \wedge (t_x \rightsquigarrow_1 t_y \vee t_x \|_1 t_y))$,
- $t_x \|_3 t_y$, iff $t_x \|_1 t_y$ and $t_x \|_2 t_y$.

We illustrate the intersection of behavioural profiles with the systems depicted in Figure 62. The lower system (c) shows a behavioural profile that corresponds to the intersection of the behavioural profiles of the upper two systems (a) and (b). Consider, for instance, transitions A and B. System (a) allows for interleaving order between both transitions, whereas system (b) is more restrictive and enforces strict order. Hence, the intersection also

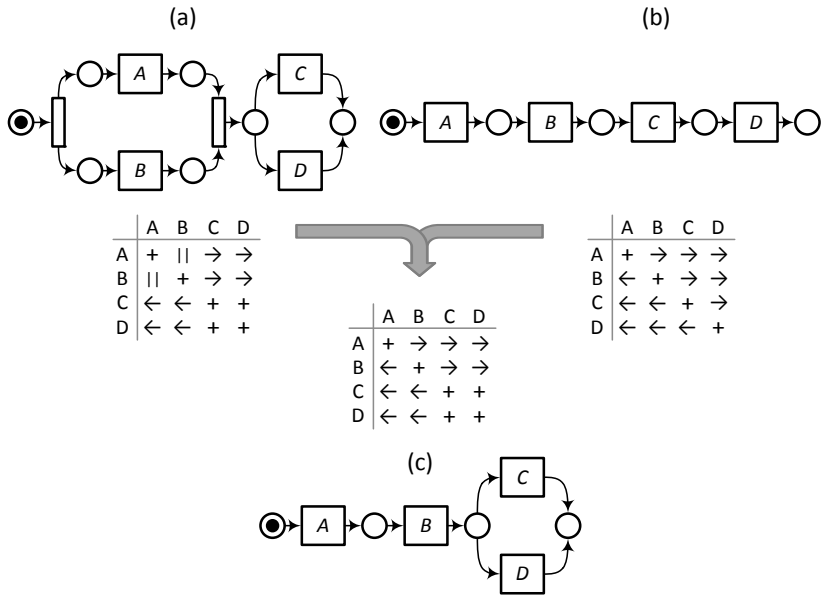


Figure 62: The behavioural profile of system (c) corresponds to the intersection of the behavioural profiles of systems (a) and (b).

defines strict order for both transitions. Due to the assumed behavioural abstraction, again, system (c) does not represent the intersection of the sets of traces of systems (a) and (b).

Referring to our example in Figure 57, we may investigate behavioural commonalities by the intersection of shared completed traces induced by the alignment. However, this intersection is empty, even if we cope with transitions that are not aligned by appropriate blocking or hiding and handle the complex correspondences by a trace partitioning, cf., the discussion in Section 6.1. The evident behavioural commonalities of both aligned net systems are not revealed by a trace-based assessment. The intersection of behavioural profiles allows us to address such scenarios.

Union. The union operation for two behavioural profiles yields a third behavioural profile that combines the weakest dependencies of two behavioural profiles given as input parameters for all pairs of transitions. We trace back the definition of the union operation to complementation and intersection using De Morgan's rule. The union $\mathcal{B}_3$ of two behavioural profiles $\mathcal{B}_1$ and $\mathcal{B}_2$ over a set of transitions T, denoted by $\mathcal{B}_3 = \mathcal{B}_1 \cup \mathcal{B}_2$ is defined as $\mathcal{B}_3 = \overline{\overline{\mathcal{B}_1} \cap \overline{\mathcal{B}_2}}$.

Properties of the Algebraic Operations

In Section 4.1, we defined behavioural profiles for net systems under the assumption of trace semantics. As a consequence, every behavioural profile shows certain properties. For instance,

strict order is antisymmetric, whereas exclusiveness and interleaving order are symmetric relations. Further, the profile relations, along with reverse strict order, partition the Cartesian product of transitions over which the profile is defined.

In the remainder of this section, we show that the essential properties of behavioural profiles are preserved by the set-theoretic operations introduced in the previous section. We start this discussion with symmetry properties of the profile relations.

Property 8.2.1. The complement, intersection, and union operations yield a behavioural profile $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ over a set of transitions T', such that relation $\rightsquigarrow$ is antisymmetric, whereas relations $+$ and $\|$ are symmetric.

The complement operation defines exclusiveness as the inverse of interleaving order, and vice versa. Strict order is obtained only for pairs in reverse strict order. Hence, the complement operation satisfies the property. The intersection operations yields exclusiveness for a transition pair if one of the profiles used as operands shows exclusiveness. Also, exclusiveness is derived if one profile shows strict order or reverse strict order, or vice versa. Intersection yields interleaving order for a transition pair if both profiles show interleaving order. Hence, the resulting exclusiveness relation and interleaving order relation are symmetric. For the case of strict order, the intersection yields an antisymmetric relation since it requires strict order in one of the operands, and either strict order or interleaving order in the other operand. Since complementation and intersection satisfy the symmetry requirements, the union operation also satisfies the property.

Using the same arguments, we conclude immediately that the operations yield a behavioural profile for which the profile relations are mutually exclusive.

Property 8.2.2. The complement, intersection, and union operations yield a behavioural profile $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ over a set of transitions T', such that the relations $\rightsquigarrow$, $+$, and $\|$ are mutually exclusive.

This property follows from the non-overlapping requirements with respect to the operand (complementation) or the two operands (intersection) when deriving the profile relation for a transition pair. Again, the property also holds for the union as the latter is traced back to complementation and intersection.

Further, the set-theoretic operations preserve the partition of the Cartesian product of transitions by the profile relations.

Property 8.2.3. The complement, intersection, and union operations yield a behavioural profile $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ over a set of transitions T', such that the relations $\rightsquigarrow$, $\rightsquigarrow^{-1}$, $+$, and $\|$ partition $T' \times T'$.

We already showed mutually exclusiveness of the profile relations. We verify the partitioning based on the completeness of the requirements with respect to the operand (complementation) or the two operands (intersection) when deriving the profile relation for a transition pair. Indeed, every transition pair that is part of the three profile relations or the reverse strict order relation is considered by the two operations. Hence, the partitioning also holds for the operation result. Again, the union operation preserves the property due to its grounding on complementation and intersection.

Finally, we consider self-relations.

Property 8.2.4. The complement, intersection, and union operations yield a behavioural profile $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ over a set of transitions T', such that any transition $t \in T'$ is either exclusive to itself, $t + t$, or in interleaving order to itself, $t \| t$.

Complementation translates exclusiveness into interleaving order, and vice versa. The intersection operation yields exclusiveness for transition pairs that are exclusive in one profile, or interleaving order for pairs that are in interleaving order in both profiles used as operands. Hence, the property holds for the complementation and intersection operations and, as a consequence, for the union operation.

8.3 MODEL SYNTHESIS FOR BEHAVIOURAL PROFILES

The algebra for behavioural profiles introduced in the previous section allows for computing with the abstracted behaviour of net systems. Explorative behavioural analysis may be supported by visualising the computation results. To this end, this section introduces a model synthesis for behavioural profiles. Given a behavioural profile that meets certain assumptions, we show how to construct a net system that shows this behavioural profile.

We first clarify the assumptions of our technique that have to be verified prior to model synthesis. Then, we present the synthesis algorithm and elaborate on its correctness.

Well-Structured Behavioural Profiles

The operations of our algebra are defined locally. They operate on a pair of transitions independent of any other transition pair. As a consequence, the operations may yield behavioural profiles that show anomalies. Dependencies defined by the behavioural profile may be contradicting, so that there does not exit any net system that satisfies all the dependencies. Whether an according net system exists depends on the applied notion of

a net system and its structural and behavioural characteristics. Consider, for instance, a cyclic strict order dependency between three transitions t_x, t_y, and t_z, i. e., $t_x \rightsquigarrow t_y$, $t_y \rightsquigarrow t_z$, and $t_z \rightsquigarrow t_x$. A sound free-choice WF-system cannot contain three transitions that show such dependencies. Using the results on the relation of the structure of sound free-choice WF-system and their behavioural profile in Section 5.1, these dependencies yield a contradiction. Once the behavioural profile is lifted to labels of transitions of a labelled net system, cf., Section 4.3, however, cyclic strict order dependencies are satisfiable. That is, there exists an according net system.

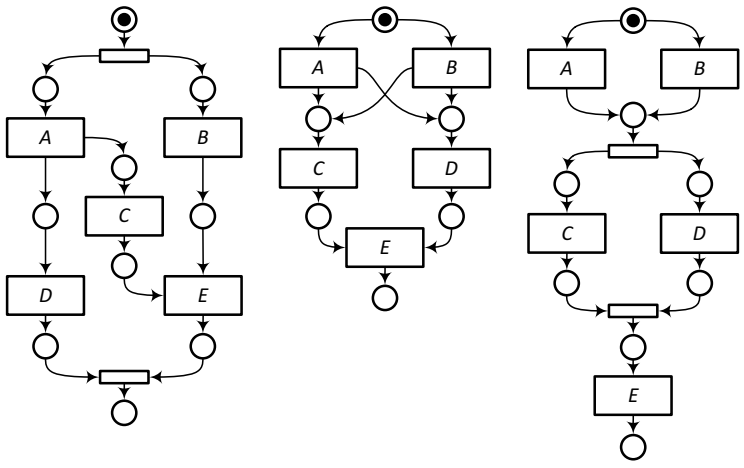
We focus on behavioural profiles that enable the synthesis of a sound well-structured free-choice WF-system. Well-structuredness refers to a net topology, in which for every node with multiple outgoing flows there is a node with multiple incoming flows and both nodes isolate a single-entry single-exit (SESE) subnet [233]. Well-structuredness turned out to be a desirable property of process models. It supports understandability [264] and enables the application of manifold analysis techniques, for instance, for the computation of temporal constraints [90]. Since many process description languages do not enforce well-structured modelling, techniques for restructuring process models have been proposed recently [361, 358].

The notion of well-structuredness is closely related to the fragments obtained by triconnected decomposition. In Section 5.2, we discussed the RPST of a net system that is derived by triconnected decomposition. Using this notion, we characterise well-structuredness as the absence of any rigid fragment in the RPST. In the following definition, we assume that the normalisation as discussed in Section 5.2 has been applied. Hence, the net system does not contain a node that has more than one incoming flow and more than one outgoing flow.

Definition 8.3.1 (Well-Structured WF-System)
Let $N = (P, T, F)$ be a normalised WF-net. N is *well-structured*, iff the set of canonical fragments of the RPST of N does not contain any rigid fragment. A WF-system $S = (N, M_0)$ is *well-structured*, iff N is well-structured.

We illustrate the notion of a well-structured WF-system with the examples in Figure 63. Systems (a) and (b) are not well-structured. Both net structures contain SESE fragments that form a rigid structure. Using restructuring techniques [361, 358], system (a) cannot be transformed into a well-structured system that is behaviour equivalent in terms of fully concurrent bisimulation [51]. In contrast, system (b) may be restructured, which yields system (c). System (c) is well-structured.

Our approach to the synthesis of a well-structured system for a behavioural profile relies on the formalism and algorithm in-



(a) (b) (c)

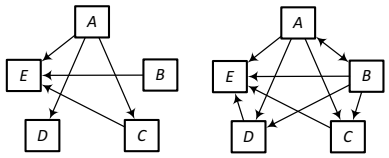
Figure 63: The WF-systems (a) and (b) are not well-structured. The WF-system (b) can be restructured, which results in the behaviour equivalent system (c).

troduced for restructuring process models in [361, 358, 362]. The restructuring techniques leverage the order relations induced by a complete prefix unfolding to guarantee the preservation of a rather strong notion of behaviour equivalence. We discussed the notion of a complete prefix unfolding as the basis for one technique to compute a behavioural profile in Section 5.3. As part of that, we established the relation between the order relations of the complete prefix unfolding and those of the behavioural profile. In the following, we show how the synthesis approach defined for the relations of the complete prefix unfolding can be transferred to the relations of the behavioural profile.

To decide whether a sound well-structured free-choice WF-system may be constructed for a behavioural profile, we need the notion of an order relations graph. It has been introduced in [358] for the order relations of a complete prefix unfolding. We adapt the notion towards the relations of the behavioural profile.

Definition 8.3.2 (Order Relations Graph)
Let $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ be a behavioural profile over a finite set of transitions T. The *order relations graph* $\mathcal{G} = (V, E)$ of $\mathcal{B}$ comprises all transitions as nodes, $V = T$, and the strict order and exclusiveness relations without self-relations as edges, $E = \rightsquigarrow \cup + \setminus id_T$.

Edges in the order relations graph stem from strict order and exclusiveness dependencies. Under the assumption that the strict order relation is antisymmetric and the exclusiveness relation is symmetric, both relations are distinguished in the order relations graph by unidirectional or bidirectional edges. Figure 64 illustrates the order relations graphs for the behavioural profiles



(a) (b)

Figure 64: The order relations graphs for the systems depicted in Figure 63. The graph (a) captures the relations of system (a), the graph (b) captures the relations for both systems (b) and (c) shown in Figure 63.

of the net systems depicted in Figure 63. As system (b) and (c) in Figure 63 are behaviour equivalent, they also have equal order relations graphs. That is due to the fact that the notion of equivalence assumed for restructuring, fully concurrent bisimulation, is stronger than behavioural profile equivalence, cf., Section 4.4.

The characteristics of a sound well-structured free-choice WF-system are related to the structure of its order relations graph. All canonical fragments of the RPST are of type trivial, polygon, or bond, see Section 5.2. Those can be characterised in the order relations graph by subgraphs that have uniform relations with all remaining nodes of the graph. To detect such subgraphs, called modules, we leverage modular decomposition [304]. This technique decomposes a graph into a rooted hierarchy of maximal non-overlapping modules. This decomposition is unique. Based on [358, 362], we formally define the concepts of the modular decomposition as follows.

Definition 8.3.3 (Modular Decomposition)
Let $\mathcal{G} = (V, E)$ be an order relations graph.

- A *module* $M \subseteq V$ is a non-empty set of nodes that have uniform relations with nodes in $V \setminus M$, i.e., $\forall\, x, y \in M, z \in (V \setminus M)$ it holds that $(x, z) \in E \Leftrightarrow (y, z) \in E$ and $(z, x) \in E \Leftrightarrow (z, y) \in E$.
- Two modules $M_1, M_2 \subseteq V$ *overlap*, iff they intersect and neither is a subset of the other.
- A module $M_1 \subseteq V$ is *strong*, iff there exists no module $M_2 \subseteq V$, such that M_1 and M_2 overlap.
- The *modular decomposition tree* is a tuple $\mathcal{MDT}_{\mathcal{G}} = (\Omega, \chi)$, such that Ω is a set of *all* strong modules and $\chi : \Omega \to \wp(\Omega)$ is a function that assigns child modules to modules, such that $\forall\, \omega, \gamma \in \Omega\ [\ (\chi(\omega) \cap \chi(\gamma) \neq \emptyset) \Rightarrow \omega = \gamma\ ]$.
- Singletons of V are *trivial* modules.
- A non-trivial module $M \subseteq V$ is *complete*, iff the subgraph of $\mathcal{G}$ induced by M is complete or edgeless. If the subgraph is complete, M is *xor-complete*. If the subgraph is edgeless, M is *and-complete*.

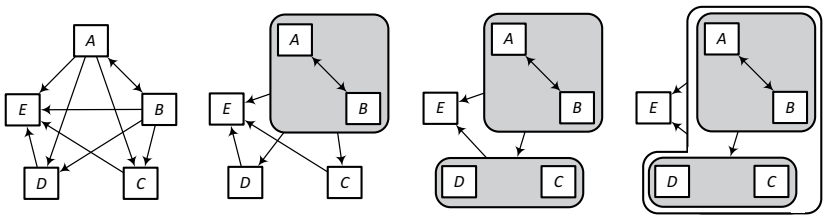


Figure 65: The step-wise modular decomposition of an order relations graph.

- A non-trivial module $M \subseteq V$ is *linear*, iff there exists a linear order $(v_1, \ldots, v_{|M|})$ of elements of M, such that $(v_i, v_j) \in E$ and $(v_j, v_i) \notin E$ for $i, j \in \mathbb{N}$, $1 \leqslant i, j \leqslant |M|$ and $i < j$.
- A non-trivial module $M \subseteq V$ is *primitive*, iff it is neither complete nor linear.

We exemplify the modular decomposition in Figure 65. Here, an order relations graph is step-wise decomposed into a hierarchy of strong modules. Two sets of nodes $\{A, B\}$ and $\{C, D\}$ are identified as strong modules that have equal relations to all other nodes in the graph. Then, both groups together form another module, as the former groups have equal relations to the node E.

The modular decomposition of an order relations graph allows for characterising behavioural profiles for which we may construct an according sound well-structured free-choice WF-system. That is, we check for the absence of a primitive module in the modular decomposition. We implicitly assume that the relational properties of a behavioural profile, e. g., with respect to symmetry of certain relations, are satisfied. At the end of the previous section, we showed that our algebraic operations preserve these properties.

Definition 8.3.4 (Well-Structured Behavioural Profile)
Let $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ be a behavioural profile over a finite set of transitions T. $\mathcal{B}$ is *well-structured*, iff the modular decomposition tree $\mathcal{MDT}_{\mathcal{G}}$ of the order relations graph $\mathcal{G}$ of $\mathcal{B}$ does not contain any primitive module.

For the aforementioned example of three transitions t_x, t_y, and t_z with $t_x \rightsquigarrow t_y$, $t_y \rightsquigarrow t_z$, and $t_z \rightsquigarrow t_x$ the profile turns out to be not well-structured. The modular decomposition of the respective order relations graph comprises a primitive module that covers all three nodes or transitions, respectively. For the three net systems in Figure 63, we depicted the respective order relations graphs in Figure 64. The graph in Figure 64a does not represent a well-structured behavioural profile since the modular decomposition tree contains a primitive module. For the graph in Figure 64b, we exemplified the modular decomposition in Figure 65. As it does not show any primitive module, the graph in

Figure 64b represents a well-structured behavioural profile. We conclude that system (a) in Figure 63 has a non-well-structured behavioural profile, whereas the profile of systems (b) and (c), both show the same profile, is well-structured.

Well-structuredness of a behavioural profile is verified efficiently using results on the operationalisation of a modular decomposition of a graph.

Proposition 8.3.1. *The following problem can be solved in linear time: For behavioural profile, to decide whether it is well-structured.*

Proof. According to Definition 8.3.4, we need to create the modular decomposition tree of the order relations graph of the behavioural profile. This is done in linear time [304]. The number of strong modules in the modular decomposition tree is linear to the size of the graph [304]. □

Finally, we show that well-structuredness of a behavioural profile is indeed a necessary condition for the existence of a sound well-structured free-choice WF-system that shows this profile. We use the notion of a WF-tree of a net system to prove this result. The WF-tree is an annotated RPST and has been defined in Definition 5.2.2.

Lemma 8.3.2. *The behavioural profile of a sound well-structured free-choice WF-system is well-structured.*

Proof. Let $\mathcal{T}_N = (\Omega, \chi, t, b)$ be the WF-tree of a sound free-choice WF-system $S = (N, M_i)$, $N = (P, T, F)$. Without loss of generality, we assume N to be pre-processed, so that each transition $x \in T$ is a boundary node of at most two trivial fragments of $\mathcal{T}_N$. Let $\alpha, \beta \in \Omega$ be trivial fragments for which the fragment entries are two distinct transitions $x, y \in T$. Since S is well-structured, the WF-tree does not contain any rigid fragment. Hence, the profile relation for x and y is deduced from the type of the lowest common ancestor (LCA) fragment, $\gamma = \mathrm{lca}(\alpha, \beta)$, and the existence of a loop fragment on the path from the root of the tree to the LCA fragment γ by Theorem 5.2.4. As such, for any fragment of the WF-tree, there is a module in the respective modular decomposition tree of the order relations graph of the behavioural profile. If the trivial fragments α and β are part of a loop fragment, the corresponding module is and-complete. If they are not part of the loop fragment, the type of the LCA fragment, $\gamma = \mathrm{lca}(\alpha, \beta)$, determines the type of the module. A polygon yields a linear module, a transition-bordered bond fragment an and-complete module, a place-bordered acyclic bond fragment an xor-complete module. Hence, $\mathcal{MDT}_S$ does not contain any primitive module and the behavioural profile is well-structured. □

Synthesis Algorithm

Once well-structuredness of a behavioural profile is verified, we proceed with the model synthesis. The synthesis algorithm iteratively constructs a net system from the modules identified by the modular decomposition. To this end, we largely rely on the synthesis algorithm presented in [361, 358, 362] for a different order relations graph.

Algorithm 6: Construction of a sound well-structured free-choice WF-system from a well-structured behavioural profile.

Input: $\mathcal{B} = \{\rightsquigarrow, +, \|\}$, a well-structured behavioural profile over a finite set of transitions T.
Output: $S = (N, M_0)$, a sound well-structured free-choice WF-system.

```
// Construct the order relations graph
1  G = (V, E) ⟵ constructOrderRelGraph(B);
// Perform modular decomposition
2  MDT_G = (Ω, χ) ⟵ modDecomp(G);
// Construct acyclic net system
3  foreach ω ∈ Ω following on a postorder traversal using χ do
4      if ω is trivial then Add transition to N;
5      if ω is and-complete then
6          Construct transition-bordered bond in N;
7      end
8      if ω is xor-complete then
9          Construct place-bordered acyclic bond in N;
10     end
11     if ω is linear then
12         Construct trivial or polygon in N;
13     end
14 end
// Normalise to get WF-structure
15 if N is missing initial or final place then
16     Add initial and / or final place to N ;
17 end
// Insert trivial circuit
18 foreach t ∈ T such that t||t do
19     Insert control flow cyclic around t in N;
20 end
21 Set M_0 as the initial marking that marks only the initial place
   of N;
22 return S = (N, M_0);
```

We outline the steps of the synthesis in Algorithm 6. First, we construct the order relations graph of the behavioural profile and perform the modular decomposition (lines 1 to 2). Then, we iterate over all identified modules to construct the skeleton

of the net system (lines 2 to 14). Trivial modules contain only a single transition, which is added to the net system. A complete module leads to the creation of a transition-bordered bond or a place-bordered acyclic bond that comprises all transitions, or subnets, respectively, that are encapsulated by the module. A linear module leads to the creation of a flow or a polygon to connect the respective transitions or subnets. The construction of all these subnets and their appropriate nesting is assumed to respect the bipartite structure of net systems accordingly.

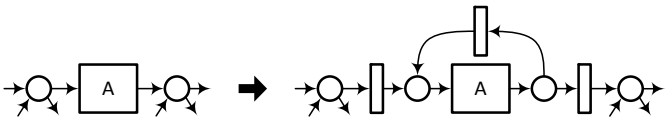


Figure 66: Insertion of a place-bordered cyclic bond for a transition with interleaving order as self-relation.

If the resulting net structure is transition-bordered, it is normalised to satisfy the structural requirements of WF-nets (lines 15 to 17). For all transitions that have interleaving order as their self-relation according to the behavioural profile, trivial circuits are inserted into the created net structure (lines 18 to 20). Those comprise a place-bordered cyclic bond and polygons. This transformation step is illustrated in Figure 66. Here, the assumption is that the construction of transition bordered bonds introduces fresh transitions as boundary nodes. Therefore, all transitions that are considered in the original behavioural profile have one incoming and one outgoing flow in the derived net structure. Finally, the algorithm sets the initial marking and returns the WF-system.

We prove correctness of the synthesis as follows. We show that the resulting net system is indeed a sound well-structured free-choice WF-system. Further, we prove that the behavioural profile used as input coincides with the behavioural profile of the created model for the respective transitions.

Proposition 8.3.3. *Algorithm 6 terminates and after termination* $S = (N, M_0)$ *is a sound well-structured free-choice WF-system that shows the behavioural profile used as input for the shared transitions.*

Proof. Termination: The set of transitions of the behavioural profile is finite, so that the order relations graph and the number of modules identified in the decomposition are finite as well. As we iterate over all transitions and all modules, the algorithm terminates.
Result: We first consider the correctness of syntax and semantics. Finally, we consider the correctness of the behavioural profile.

- Syntax: The algorithm creates a net system $S = (N, M_0)$ by iterating over all modules following on a postorder traversal

of the modular decomposition tree. Hence, for all transitions and places there is a path from (to) the node that represents the entry (exit) of the fragment created for the root module. If those nodes are transitions, the algorithm adds an initial and a final place. Hence, N shows workflow structure. The algorithm constructs only trivial, polygon, and bond fragments (the trivial circuit is a bond as well). Further, the insertion of trivial circuits ensures that the transition in the post-set of the exit of the cyclic bond has no other places in its pre-set (if this transition would be the exit of a transition-bordered bond, the net would be non-free-choice). Hence, N is also free-choice.

- Semantics: The net shows workflow structure, is constructed by nesting trivial, polygon, and bond fragments, and the initial marking marks the initial place. The bond fragments constructed for modules are acyclic, either place or transition-bordered. The construction of the trivial circuit inserts polygons and a cyclic place-bordered bond. Trivial and polygon fragments cannot cause unsoundness and the created bonds satisfy the requirements for soundness discussed in Lemma 5.2.1 and Lemma 5.2.2. Hence, S is sound.
- Correctness of the Behavioural Profile: The constructed net system $S = (N, M_0)$, $N = (P, T, F)$, is sound well-structured, and free-choice. By Lemma 8.3.2, the behavioural profile of S is well-structured. That this behavioural profile coincides with the behavioural profile used as input for the algorithm follows from the types of the constructed fragments. Neglecting the trivial circuits inserted at the end, the algorithm creates a trivial, polygon, or bond fragment depending on the type of the module, i. e., depending on the relations observed between the nodes (transitions) of the module in the order relations graph. For two distinct transitions $x, y \in T$, this fragment is the lowest common ancestor (LCA) of the trivial fragments $\alpha, \beta \in \Omega$ for which the fragment entries are x and y, respectively. Neglecting the trivial circuits, the type of the LCA fragment determines the profile relation by Theorem 5.2.4. Insertion of trivial circuits does not impact on the relation between two distinct transitions, as a circuit comprises only one transition that is part of the original behavioural profile. Still, it leads to interleaving order as the self-relation by Theorem 5.2.4 for such a transition. As such a circuit is introduced only for transitions with interleaving order as the self-relation, the behavioural profile of the constructed net system coincides with the behavioural profile used as input for the algorithm for the shared transitions.

□

Corollary 8.3.4. *The following problem can be solved in linear time: Given a well-structured behavioural profile, to construct a WF-system that shows the behavioural profile.*

Proof. We assume all relations used in the algorithm to be represented by their characteristic functions, i. e., to be encoded as bi-dimensional arrays that map to either zero or one. Then, adding an entry to a relation and checking membership for a tuple takes constant time. The order relations graph is built in linear time to the size of the behavioural profile. The modular decomposition tree of this graph is obtained in linear time either [304]. Further, we iterate over the number of strong modules in the modular decomposition tree, which is linear to the size of the graph [304]. Construction of the respective subnets takes linear time to the size of the behavioural profile. Normalisation requires the check for initial and final places, which also takes linear time. Insertion of the trivial circuits takes linear time to the size of the behavioural profile. □

After we studied the relation between behavioural profiles and the existence of a sound well-structured free-choice WF-system, we conclude the following.

Theorem 8.3.5. *There exists a sound well-structured free-choice WF-system, if and only if, the behavioural profile is well-structured.*

Proof. $\Rightarrow$ follows from Lemma 8.3.2, $\Leftarrow$ from Proposition 8.3.3. □

8.4 APPLICATION

In this section, we discuss how the set algebra and the synthesis for behavioural profiles are applied to address the questions raised for explorative behavioural analysis in Section 8.1. Let $\mathcal{B}_1, \ldots, \mathcal{B}_n$ be a set of behavioural profiles of aligned process models that have been normalised for the aligned activities.

C1: Shared behaviour: Greatest Common Divisor. The shared behaviour may be referred to as the *Greatest Common Divisor* (GCD), cf., [449]. Given a set of aligned process models, the GCD is characterised by a behavioural profile $\mathcal{B}_{GCD}$ over the aligned activities. It is derived by computing the intersection of all profiles $\mathcal{B}_1, \ldots, \mathcal{B}_n$. Hence, the GCD integrates the dependencies shared by all aligned models. The profile $\mathcal{B}_{GCD}$ may be checked for emptiness. If it is empty, all aligned process models impose contradicting dependencies for the aligned activities. If the profile $\mathcal{B}_{GCD}$ is well-structured, we may synthesise a process model that represents the GCD. Further, the behavioural commonality between a process model and the GCD may be quantified using the degree of behavioural profile consistency defined in

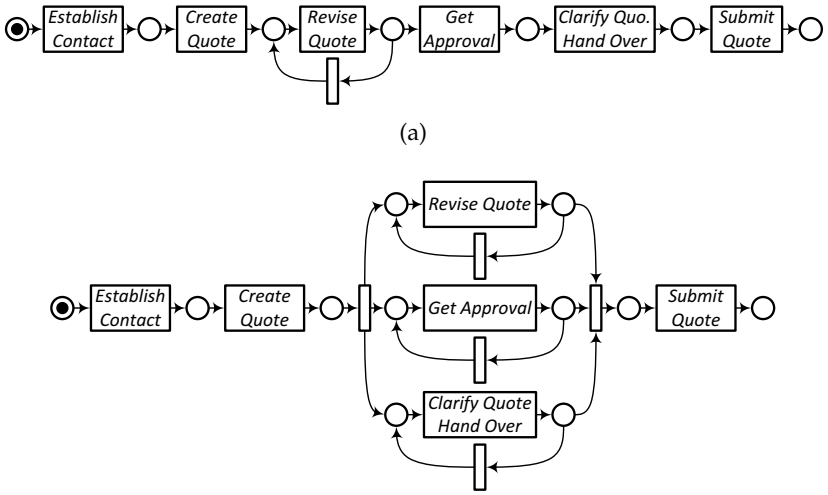


Figure 67: Net system (a) represents the GCD for the aligned net systems of Figure 57, net system (b) represents the respective LCM. Both systems have been reduced by a *fusion of series places* [329]. Further, we took the labels of system (a) in Figure 57 for the GCD and LCM transitions.

Section 7.2. Then, this degree quantifies how much additional behaviour the process model allows for, relative to the behaviour shared with all aligned process models.

Figure 67a illustrates the GCD for the aligned net systems of our initial example, the lead-to-order process illustrated in Figure 57. To be able to compute with the behavioural profiles of the net systems depicted in Figure 57, those have to be normalised according to the transition sets that are part of complex correspondences. Taking a normalisation threshold of one, for instance, we identify strict order as the dominating relation between the two transitions 'Publish Quote' and 'Delegate Quote Handling', and transition 'Send Quote' in system (b) in Figure 57. For the normalised behavioural profiles, we computed the intersection and synthesised a net system. The behavioural profile of the GCD comprises all behavioural dependencies from system (b) in Figure 57, as they are more restrictive than those imposed by system (a). The net system depicted in Figure 67a, slightly reduced using a *fusion of series places* [329], has been synthesised from the obtained behavioural profile. Hence, it visualises the basic order dependencies of both aligned net systems. Even though these dependencies all stem from system (b) in Figure 57, the system in Figure 67a is not identical to the former. Neglecting the differences related to the normalisation and the transition labels, we observe that there are differences in the specified causal dependencies. As an example, the causal dependency between transitions 'Establish Contact' and 'Create Quote' in the GCD does not hold between the corresponding transitions

in system (b) in Figure 57. This difference is due to the behavioural abstraction induced by behavioural profiles.

C2: Most general behaviour: Least Common Multiple. The most general behaviour is referred to as the Least Common Multiple (LCM) of a set of aligned process models, see also [449]. It is characterised by a behavioural profile $\mathcal{B}_{LCM}$ over the aligned activities that is derived by computing the union of all profiles $\mathcal{B}_1, \ldots, \mathcal{B}_n$. The LCM imposes the weakest dependencies found in the set of aligned models for a pair of activities. Again, the LCM is visualised by synthesising a process model from $\mathcal{B}_{LCM}$ if the respective requirements are met.

For our example alignment, Figure 67b illustrates the LCM. We normalised the behavioural profiles of the systems in Figure 57 as discussed earlier and computed the union of these profiles. Then, we synthesised a net system to visualise the LCM. The concurrent enabling of three transitions is caused by the interleaving order imposed by the LCM. As all of them may occur multiple times – interleaving order is the self-relation – they are also part of trivial circuits. The system in Figure 67b, again, has been reduced by a *fusion of series places* [329].

C3: Shared forbidden behaviour: Complement of the LCM. To characterise the behaviour that is forbidden by all aligned process models, a behavioural profile $\mathcal{B}_{SFB}$ over the aligned activities is derived as the complement of the LCM, $\mathcal{B}_{SFB} = \overline{\mathcal{B}_{LCM}}$. However, this profile does not directly capture all dependencies that are not implemented in any process model due to the strictness of behavioural relations discussed in Section 8.2. Conclusions are drawn solely from the (reverse) strict order and interleaving order relations of the profile $\mathcal{B}_{SFB}$. If $\mathcal{B}_{SFB}$ defines interleaving order between two activities, all aligned process models show exclusiveness for these activities. Hence, the potentially arbitrary order implied by the interleaving order dependency is forbidden in all models. Similar conclusions are drawn for (reverse) strict order dependencies in $\mathcal{B}_{SFB}$.

We illustrate this concept with the transitions 'Revise Quote' and 'Get Approval' of our initial example. For both transitions, the LCM defines interleaving order, which yields exclusiveness in the complement. As exclusiveness is the strictest relation, we cannot draw any conclusions on shared forbidden behaviour. For the transitions 'Create Quote' and 'Get Approval', the LCM defines a strict order dependency from the former to the latter. Therefore, the reverse strict order dependency in the complement is shared forbidden behaviour. The occurrence of the transition 'Get Approval' *before* the transition 'Create Quote' is not allowed in any of the aligned net systems.

C4: Redundancy of specified behaviour: Inclusion of behavioural profiles. Given the behavioural profiles $\mathcal{B}_1$ and $\mathcal{B}_2$ of two aligned

process models, the question whether the behaviour defined by one model is captured in the other model for the aligned activities is traced back to the inclusion of their behavioural profiles, i. e., $\mathcal{B}_1 \subseteq \mathcal{B}_2$. In this case, all dependencies imposed by the first model would be equal or less strict than the dependencies imposed by the second model. As a consequence, the behaviour of the latter model, according to abstraction assumed by behavioural profiles, is covered by the former model. That suggests investigating whether the two process models are indeed required to serve the respective modelling purposes. It may be possible to integrate the behaviour defined by subsumed model into the subsuming model to reduce the number of process models capturing the same business process.

Investigating the two net systems given in Figure 57, the behavioural profile of system (a) includes the profile of system (b) for the aligned transitions, once normalisation has taken place. Hence, when aiming at reducing the number of process models that depict the lead-to-order process, the existence of model (b) may be challenged.

8.5 EXPERIMENTAL EVALUATION

As an evaluation, we review the models of the SAP reference model [95]. Even though the models of this collection capture different functional aspects of an enterprise, they are not orthogonal. There are rather large clusters of models that show a functional overlap. For pairs of models that overlap, we constructed an alignment and investigated its consistency in Section 7.3. Based on the obtained results, we argued that behavioural profile consistency is particularly suited for measuring consistency in this setting.

In this section, we go one step further and report on an exploration the behavioural commonalities of clusters of aligned process models. This analysis relies on the set algebra and the synthesis for behavioural profiles. For each cluster, we computed the most general behaviour, the Least Common Multiple (LCM) of the aligned process models. In addition, we investigated behavioural subsumption relations among the models.

In the remainder of this section, we summarise the experimental setup, present the obtained consistency results, and discuss the results.

Experimental Setup

For our analysis, we used the subset of models that was also selected for the experiment on the computation of behavioural profiles in Section 5.4. This subset contains all models that are

Table 5: Clusters of aligned net systems stemming from the SAP reference model.

# Min Correspondences	4	6	8	10	12	14	16	18	20	22	24	26
# Clusters	84	48	33	23	23	21	15	11	9	2	1	0
Avg Size of Clusters	3	3	3	3	3	2.6	2.4	2.3	2.3	2	2	0
Max Size of Clusters	10	9	8	8	7	6	4	4	4	2	2	0
# Systems in Clusters	212	124	88	63	56	47	36	25	21	4	2	0
# Subsumed Systems	127	73	55	41	35	28	20	14	12	2	1	0

non-trivial, free of syntax errors and instantiation issues, normalised with respect to their entries and exists, and free of behavioural anomalies. The subset comprises 493 models that we transformed into sound free-choice WF-systems. Our transformation ensures that each EPC function and EPC event is represented by a single transition.

As for the experiment on the quantification of consistency, we constructed alignments for groups of net systems with elementary correspondences between transitions representing EPC nodes with equal labels. We proceeded step-wise by increasing a threshold for the number of correspondences that minimally has to hold between all pairs of net systems in a group of models, referred to as a cluster. For all systems that are part of at least one cluster, we computed behavioural profiles using the techniques introduced in Section 5.1 and Section 5.2. Using the behavioural profiles, we then computed the LCM for each cluster.

Experimental Results

Table 5 gives on overview of the observed clusters of aligned net systems that stem from the SAP reference model. Given a threshold for the required number of elementary correspondences (# *Min Correspondences*), Table 5 depicts the number of clusters, their average and maximal size, and the number of net systems that are part of the clusters. For instance, requiring the net systems of a cluster to be aligned by at least 12 correspondences leads to 23 clusters, which comprise 56 distinct net systems. The average size of these clusters is three net systems, the maximum size is seven systems.

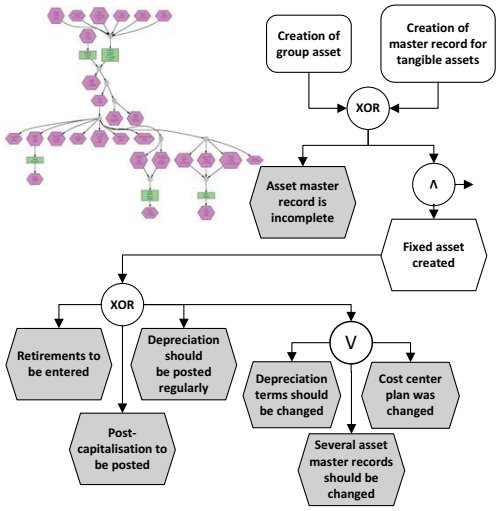
To explore the commonalities for all identified model clusters, we derived the LCM from the behavioural profiles of the respective net systems. A manual inspection revealed that the LCM was often equal to the profile of at least one system in the cluster. Based on this observation, we checked for all clusters whether

the behavioural profile of one net system includes the profiles of all other systems for the aligned transitions. This was the case for all but two clusters. The significance of this observation is also indicated by the absolute number of concerned net systems. The last row of Table 5 presents the absolute number of net systems for which the behavioural profile is subsumed by the profile of another net system in the same cluster.

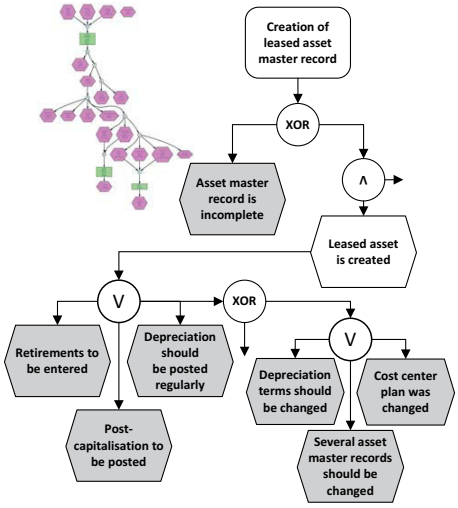
Discussion of the Results

The structural clustering of process models in the SAP reference model reveals that a large number of models show a functional overlap. As discussed in Section 7.3, we have no information on the reasons for this observation. Also, we do not know whether the overlapping process models represent different perspectives of a common business process or different variations of the process. For an evaluation of behaviour consistency of correspondences between equally labelled EPC nodes, however, this question is of minor importance.

In any case, the presented approach helps to investigate behavioural commonalities. For the given model collection, the computation of the LCM for each model cluster revealed various behavioural subsumption relations between models, once the behaviour is abstracted by behavioural profiles. For illustration purposes, Figure 68 depicts excerpts of two EPC models. Both models are aligned by 16 elementary correspondences between EPC nodes with equal labels, some of them are highlighted in the excerpts. For the nodes that are part of correspondences, both process models impose different behavioural dependencies. Still, the behaviour of the model in Figure 68a includes the behaviour of the model in Figure 68b in terms of the profile relations for pairs of corresponding EPC nodes. The question whether both process models are still required to capture the respective operations cannot be answered for the general case. Still, the detection of subsumption relations points to process models that may be integrated. Then, the number of related models that capture the same business process may be reduced. This ultimately lowers the effort needed to construct alignments between models in this collection and to evaluate their behaviour consistency. Also, our model collection is a reference model. Avoidance of redundancy in the specified behaviour can be seen as a quality criterion for the model collection.



(a)



(b)

Figure 68: Excerpts of two process models in the SAP reference model.

8.6 RELATED WORK

We motivated the definition of algebraic concepts based on behavioural profiles with the need to explore commonalities and differences. In Section 8.1, we discussed related work on algebraic concepts based on behaviour equivalences. Here, work on behaviour inheritance [138, 451, 33, 412] may be seen as a notion of inclusion of behaviour. View integration [366, 343, 314] and process model merging [330, 180, 258, 259] provide solutions to the union of behaviour. As for a comprehensive algebra for process models, existing work is restricted to the syntax of net systems [343]. Hence, despite the aforementioned results, there is no overarching behavioural algebra.

Our general idea of defining a set algebra for computing with behaviour is inspired by algebraic concepts for operating guidelines [228]. Operating guidelines are a formal concept for the description of interaction behaviour. Even though this work also defines set-theoretic relations and operations for a behavioural model, the focus is on automata that describe interaction protocols. Further, the goal of this algebra is to reason on whether two services may interact successfully. In contrast, we aim at the explorative analysis of behavioural commonalities and differences of process models.

Our work also relates to techniques for diagnosing process model differences. Such diagnosis is an integral part of process model change management. We reviewed the according related concepts in Section 7.5.

In the remainder of this section, we review techniques for modelling process variability. Finally, we discuss work on process mining, which is close to the presented model synthesis from a behavioural profile.

Modelling Process Variability

Process models that capture the same business process, but have been created for different purposes are closely related to process variants. Although there is a fundamental difference between the two settings – either we refer to the same original or to different originals – the distinction is often blurred in practise. Actually, the relations between the respective process models are very similar in both cases. The same holds true for the techniques to detect, quantify, or manage process model differences.

Variability of business processes may be addressed at design time or at run time. The former controls variability by explicitly considering it in the course of process modelling. The latter refers to flexible and adaptable process automation. There is a large body of work on flexible and adaptable process automation [375, 400, 454, 376, 383, 488, 287, 97]. An overview can be found in [489]. The concepts introduced in this chapter are not concerned with the automation of business processes. Hence, we limit the discussion to concepts for modelling variability during design time.

Approaches to modelling process variability are inspired by work on modelling variability in Software Engineering [28]. In particular, mechanisms from Software Product Line Engineering (SPLE) [356, 35, 185] have been adapted. Most approaches for modelling process variability incorporate the notion of a configurable reference model. This model is configured by means of well-defined operations to obtain a model for a process variant. As for these operations, process variants may be derived

from reference models through model projection [41, 379] or explicit configuration mechanisms. Such mechanisms that extend process description languages with configuration capabilities have been presented for EPCs [392, 389, 379], YAWL [181], BPEL [181], UML activity diagrams [96, 407, 371], BPMN [407], WF-systems [461], and well-structured process models within the Provop framework [191, 193].

Configuration mechanisms allow for marking activities to be excluded or to be skipped under certain conditions that are determined at run time [392, 389]. Control flow routing elements may be also be subject to configuration to restrict the possible behaviour towards the process variant [392, 389]. Also, configuration of a reference model has been tackled based on blocking and hiding of activities as known from behaviour inheritance [461, 181]. Inspired by the stereotyping functionality of object oriented modelling languages, stereotypes that mark variation points have been proposed for BPMN process models and UML activity diagrams [407, 371]. Then, configuration is grounded on these variation points, e. g., optional activities or alternative activities, or activities that extend the functionality of the default activity may be plugged into a variation point. In addition, branching conditions can be parameterised as part of the configuration [407]. Similarly, complete subgraphs of UML activity diagrams may be annotated according to their presence in a certain process variant [96]. Configuration of reference models within Provop [191, 193] is done differently. Provop relies on a catalogue of change operations that may be applied to a reference model, see [488] for an overview of these patterns.

Since explicit configuration mechanisms increase the complexity of a process model, several approaches also provide support for the configuration of a reference model. For instance, configuration may be guided by questionnaires that, once completed, induce a set of configuration operations [389]. Other work guides model configuration by product or service hierarchies [379]. Feature diagrams [408] known from SPLE have also been used as a means to control process model configuration [407]. Further, many of the aforementioned approaches allow for modelling dependencies between configuration options. For instance, configurations operations may be causally related or mutually exclusive.

Similar to the properties that are preserved when generating a member of a product line through refinement, cf., [36], most process model configurations also define structural and behavioural properties that are preserved. Here, the soundness property has been in the centre of interest. Approaches that guarantee soundness of the process variant obtained through configuration have

been proposed for WF-systems [461], configurable EPCs [464], and within the Provop framework [192].

Many of the discussed approaches incorporate algorithms to construct a reference model from a given set of models representing process variants. Besides the aforementioned work on process model merging [330, 180, 258, 259], the MinAdept project deserve further discussion [272, 273, 275]. Taking the same notion of well-structured process models as the Provop framework, a configurable reference model is mined from a set of process variants. This approach is based on order matrices. In Section 4.5, we discussed that the behavioural profile can be seen as a generalisation of such an order matrix. Order matrices of process variants can be combined into an aggregated matrix that associates weights to the entries of the behavioural relations. Using these concepts, two approaches may be used to come to a reference model. First, an existing process model is used as a start and heuristic search algorithms are used to find adaptations of its order matrix so that soundness and well-structuredness are preserved and a change edit distance to all variants is minimised [273]. An alternative approach computes an aggregated order matrix from all matrices of process variants [272, 275]. Then, activities are step-wise clustered and the matrix is adjusted accordingly until a complete model is synthesised. As such, this approach works similar as computing a union of behavioural profiles and then applying the presented model synthesis. In contrast to our union operation for behavioural profiles that is grounded on a notion of strictness of behavioural relations, the aggregation of order matrix assigns weights to the different observed relations. Further, we provide a characterisation of the existence of a process model based on the behavioural relations. Despite these differences, the stepwise clustering within the order matrix performs a modular decomposition that is also leveraged in our approach.

In conclusion, work on process model variability offers means to control the creation of process models representing different variations of a business process. These techniques may in principle be used to control the creation of process models that capture the same business process for different purposes. As discussed for process model views in Section 6.4, however, it is at least questionable whether the variety of process modelling drivers can be accommodated by such a controlled approach. Therefore, our work takes a different approach by offering the freedom to create process models independently at the expense of investigating their consistency a posteriori. Instead of identifying and explicitly representing the deviations between related process models, we compute their commonalities and differences once an alignment is established with the proposed set algebra.

Hence, we do not assume that deviations between two process models can be traced back to variability mechanisms of a common root process model. Our algebraic operations are applicable in the general case. However, we compute with abstracted behaviour whereas a configurable reference model is commonly assumed to subsume all traces of all possible process variants.

Process Mining

The developed method for the construction of a net system from a behavioural profile extends the family of process model synthesis techniques. Process mining aims at the construction of a process model from event logs, i. e., observed execution sequences [457, 460, 446]. Process mining received much attention in the last decade. A large number of process mining techniques and concepts supporting Business Intelligence have been presented, see [446] for an overview of the field.

Several process mining algorithms take a relational approach, e. g., those proposed in [11, 99, 510, 457, 459, 511, 106]. That is, the construction of a process models builds upon behavioural relations that are derived from event logs. These relations are close to those of the behavioural profile. Since we have discussed the commonalities and differences of these relations in detail in Section 4.5, we focus on the operationalisation of the synthesis at this stage.

The relations of the α-algorithm rely on direct successership of transition occurrences [457, 446]. This local perspective enables the detection of certain control flow patterns. For instance, a conflict between two transitions B and C following another transition A is typically manifested in the relations of the α-algorithm in terms of causality, $A \rightarrow B$ and $A \rightarrow C$, and the absence of any direct successership for transitions B and C, B#C. Given such a combination of local dependencies between the transitions, the α-algorithm constructs the place and the flows implementing the conflict. Extensions of the α-algorithm, the α^{+}-algorithm [104, 105] and the α^{++}-algorithm [512], are based on the same basic principle. In addition, these algorithms apply preprocessing of event logs and post-processing of the created model, or refine and extend the set of behavioural relations. More advanced mining algorithms, such as the heuristics miner [510, 511] or the fuzzy miner [189], also follow this idea. To cope with log incompleteness and noise, these approaches take frequencies of the observed relations into account and apply abstraction and aggregation heuristics to the observed events.

The synthesis presented for behavioural profiles works differently. Instead of identifying local dependencies that translate directly into the net structure, we seek for groups of transitions

that translate into well-structured subnets. Hence, conceptually, our approach is similar to work on region-based approaches to model synthesis. Once a state space has been constructed from observed execution sequences, regions of states that are encapsulated in terms of entering, exiting, and contained transitions are identified [141, 92, 83]. Then, a net system is constructed based on these regions. The work reported in [466, 471] adapts these ideas for process mining. The construction of a state space from an event log may involve abstractions to achieve generalisation [466]. Also, an iterative construction of a state space for a large event log has been proposed [471]. Further, regions may also be identified for the language induced by the observed execution sequences. Then, an initial net system comprising all transitions, but no places, is step-wise extended with places to restrict the behaviour such that the system still allows for the observed behaviour [45]. Following this idea, advanced methods for the selection of such places have been presented in [469].

The presented operationalisation of model synthesis is motivated by the ability to decide on the existence of a sound net system, which shows the respective behavioural profile, based on the profile relations. The characterisation of the existence of a net system can be decided efficiently. On the downside, this characterisation is restricted to a dedicated class of net systems, sound well-structured free-choice WF-systems. Arguably, soundness is a desired property for net systems representing business processes. Enforcing well-structuredness and free-choiceness, in turn, have to be seen as limitations of the presented synthesis. To overcome these restrictions, one may rely on the α-algorithm. Following on the discussion presented in Section 4.5, the relations of the α-algorithm may be derived from the relations of the behavioural profile. Since the behavioural profile is a behavioural abstraction, derivation of the α-algorithm involves various design decisions. As an example, consider three transitions $\{A, B, C\}$ that are in strict order, $A \rightsquigarrow B$, $A \rightsquigarrow C$, $B \rightsquigarrow C$. The strict order relation may directly be used as the causality relation of the α-algorithm. This yields a net system in which each transition in the sequence may be bypassed by firing a silent transition. However, we may also apply a transitive reduction first. Then, only the relation obtained by transitive reduction is used as the causality relation of the α-algorithm. In this case, we would end up with a net system, in which an occurrence of transition A cannot be followed directly by an occurrence of transition C.

Although it would be possible to rely on the α-algorithm, we are not aware of any characterisation of the existence of a net system based on the relations of the α-algorithm, or one of its extensions. This is problematic in our setting. We want to en-

sure that the net system synthesised from a behavioural profile indeed shows this profile. If it cannot be decided a priori whether there exists a respective net system, synthesis first has to be conducted before the behavioural properties are verified. Then, one may chose one of the options to derive the relations of the α-algorithm from the behavioural profile and check the synthesised model for soundness and adherence to the behavioural profile used as input. Constructing a net system first (may be multiple times) and then examining the profile of the constructed system, apparently, increases the required computational effort for the synthesis.

In the line of such a *trial and error* approach, one may rely on the principles of genetic mining [459, 106]. A set of candidate net systems may evolve by crossover and mutation operations. These operations are performed on the systems that show the best approximation of the behavioural profile for which a model should be synthesised. To quantify the closeness of two behavioural profiles, one may rely on the consistency measures defined in Section 7.2. In this way, a system that is close to the profile used as input may be derived.

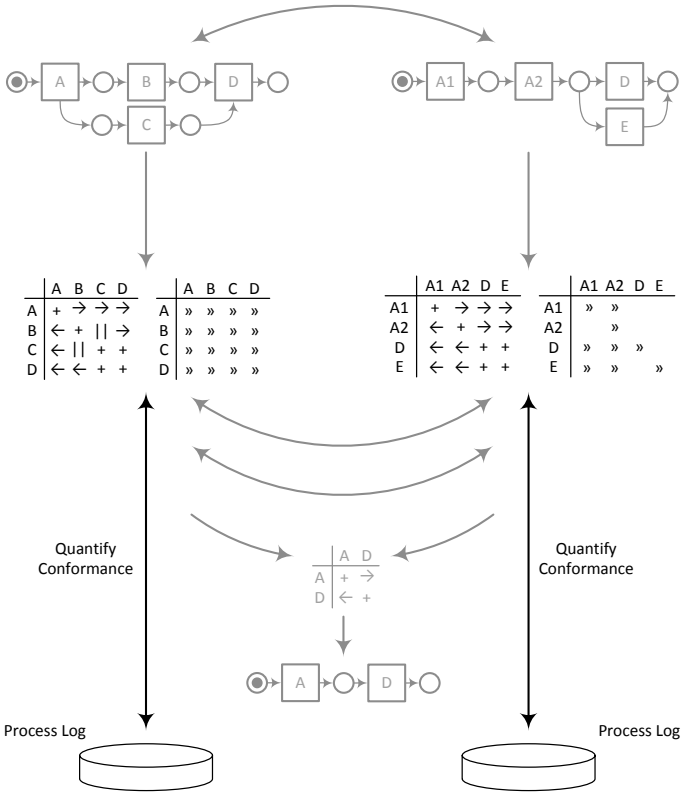
8.7 CONCLUSION

Consistency notions and measures are at the core of any analysis of behaviour consistency of an alignment between process models. Still, the analysis may be supported by means to explore behavioural commonalities and differences. This allows us to interpret the results and ensures traceability of the obtained consistency values. In this chapter, we illustrated this need by a set of exemplary analysis questions. Then, we introduced a set algebra based on behavioural profiles. It allows us to compute with abstracted behaviour using set theoretic relations and operations. As an auxiliary concept, we introduced a model synthesis from a behavioural profile. We provided a characterisation of the existence of a sound well-structured free-choice WF-system based on the behavioural profile and introduced a synthesis algorithm. Further, we showed how the set algebra and the model synthesis are utilised to answer the analysis questions. Finally, we reported on findings of applying our algebraic operations for an industry model collection.

The presented concepts enable multifaceted behavioural analysis. We restricted the discussion to a few exemplary analysis questions. Having defined a comprehensive set algebra for the abstracted behaviour of net systems, however, we are able to answer all analysis questions that can be traced back to essential algebraic concepts.

9 ANALYSING LOG CONFORMANCE

This chapter is based on results published in [503, 507].



In the previous chapters, we concentrated on behaviour consistency of process models. Conclusions drawn on the model level raise the question of how well the models actually represent the abstracted original. If the original, the business process, actually exists, this question may be approached on the basis of observed execution sequences. Then, consistency between the behaviour as defined by the model and as observed in reality can be evaluated. However, the observed behaviour should not be taken as the original. Actually, it also provides an abstracted view on business operations in the sense of a mapping and a reduction of the business process. There may be even different sources of observed behaviour for a common business process, which yield different models of observed behaviour.

In this chapter, we evaluate the consistency of observed behaviour with respect to a process model. This evaluation is known as conformance analysis of process logs. A log can be

seen as a model of observed behaviour. Hence, we leverage the same formalism used for evaluating consistency between process models. This insures a uniform consistency evaluation for all behavioural models of a common business process.

This chapter is structured as follows. In Section 9.1, we give background details on analysing conformance of observed behaviour. Then, we define behavioural profiles for process logs in Section 9.2. Using behavioural profiles of process models and logs, we propose conformance measures in Section 9.3. These measures allow for quantification of conformance. In practise, feedback on non-conformance is crucial to be able to interpret the obtained conformance values. We address this need in Section 9.4. We provide means for root cause analysis for single observed execution sequences, and on the level of a complete log. All proposed concepts are applied in a case study using real-world log data in Section 9.5. Finally, we review related work in Section 9.6 and conclude in Section 9.7.

9.1 CONFORMANCE ANALYSIS

Conformance analysis received considerable attention in recent years. Various drivers, among them the increasing importance of compliance management, led organisations seek for a better control of their processes. To this end, detection of non-conforming processing is an essential step. Conformance analysis assumes the existence of a process model that captures the *expected* behaviour. Further, information on the actual processing must be available in the form of a process log (or log). A log comprises cases that represent the *observed* behaviour in terms of execution sequences of activities of the respective process model. Extracting events from an IT-system and relating them to activities of a process model may be a cumbersome task. Depending on the logging facility, events may have to be filtered and aggregated, or generated based on state changes on the database level, see [413].

In this section, we first discuss different dimensions of conformance analysis and introduce an example process model and log. Then, we review conformance measurement based on log replay, before we turn the focus on the application of behavioural relations for conformance assessment.

Analysis Dimensions

Once a process model and a process log is available, their relation is evaluated along a set of dimensions [396, 492]. *Fitness*, also referred to as *recall*, measures to what degree the behaviour of a single case (or a complete process log) is captured in a process model. Other dimensions focus on the appropriateness of

a process model with respect to a process log. In this case, the degree to which the process model is restricted to the observed behaviour (*precision*) or allows for additional behaviour (*generality*) is quantified [466].

We focus on the fitness dimension. Fitness measures provide feedback on cases that do not conform to the process model and quantify any behavioural deviation. Precision and generality, in turn, aim at quantifying the quality of a process model with respect to the observed behaviour. Hence, these metrics are mainly used to judge on the quality of process models that are discovered by mining algorithms [395].

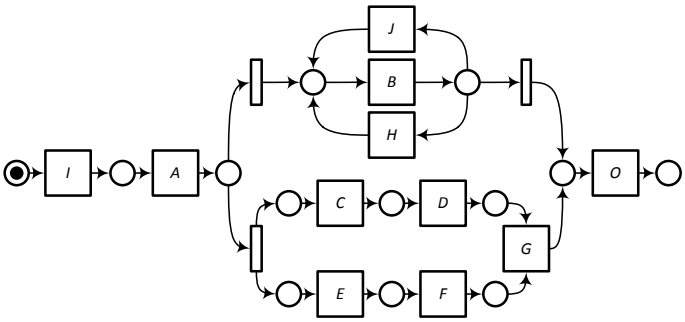


Figure 69: Example net system for conformance analysis.

We illustrate conformance measurement, in the sense of fitness measurement, using the example process model depicted in Figure 69. In a perfect setting, the cases of a log comply with the behaviour defined by the process model. Then, the cases are *valid* execution sequences, aka traces. In practise, however, observed execution sequences often deviate from the predefined behaviour. This may be a problem of the model when it does not meet validity and completeness requirements [278]. When the process model has a normative character, deviations may be caused by information systems that record a log but do not explicitly enforce the execution order of activities. It is also possible that people deliberately work around the system [460]. This may result in cases for the process model in Figure 69, such as the following.

- Case $c_1 = \langle I, A, E, C, D, F, G, O \rangle$
- Case $c_2 = \langle I, A, C, B, G, F, O \rangle$
- Case $c_3 = \langle I, A, B, J, H, B, O, G \rangle$
- Case $c_4 = \langle I, C, E \rangle$
- Case $c_5 = \langle F, C, D, G \rangle$

From these five cases, only the first one is also a valid execution sequence of the net system. Still, the other cases capture a certain share of the behaviour defined in the system. Such cases make it necessary to measure conformance a posteriori.

Conformance based on Replay

Existing conformance measures mainly rely on state-based techniques that involve replaying the cases of a process log, i. e., the single observed execution sequences [183, 511, 395, 107, 176]. Then, conformance may be computed as the share of cases of a process log that can be replayed in the process model [183, 511]. More fine-granular measures try to replay a case step-wise in the process model and quantify the number of execution steps that are in line with the process model semantics [395, 107, 176].

The basic idea behind the classical fitness measure [107] can be summarised in Petri net terms as follows. We count transitions that are enabled in the model when they appear in the case and relate them to the overall number of transitions. If a transition is not enabled but occurs in the case, it is forced to fire, which produces a token on each of its output places. Then, we quantify conformance of a case against the process model as a ratio of enabled transitions to the total number of transitions. For the system in Figure 69, case c_1 can be replayed and, therefore, has a conformance value of one. Instead, case c_2 can be replayed until transition B appears in the case. It is then fired without being enabled, which also holds for transitions G and F. Therefore, four firings are conforming out of seven firings altogether, yielding a fitness value of 0.57. This approach leads to conformance values of 0.63 for case c_3, 0.33 for case c_4, and 0.5 for case c_5.

Quantifying conformance based on log replay has to cope with certain challenges. If a transition is not enabled in a certain marking, it has to be checked whether it may be enabled by a firing a sequence of silent transitions. This increases the needed computational effort, as a certain share of the state space of the system has to be explored [395]. In addition, silent transitions may lead to conformance values below one for valid execution sequences, which has to be addressed separately [6].

Conformance based on Behavioural Relations

As for consistency assessment between two process models, relational semantics may be used to judge conformance of a log. Again, different sets of behavioural relations may be selected for this kind of analysis. We address conformance analysis of process logs as part of a uniform framework for consistency evaluation. Conceptually, this framework treats all behavioural models, may they be process models or process logs, in the same way. By leveraging the same formal grounding, behavioural profiles, the results are comparable to a large extent, as biases caused by varying expressiveness of the formalism are avoided.

Conformance measurement using behavioural profiles works as follows. For all pairs of activities, we evaluate to which extent the order dependencies of the behavioural profile and the co-occurrence relation of the causal behavioural profile are respected in a certain case. The ratio of relations obtained for the process model that coincide with (or subsume, as we will discuss later) the relations obtained for a case to all relations is used to quantify conformance. Based on the conformance values obtained for single cases, conclusions on the conformance of a log can be drawn. This approach works efficiently. Behavioural profiles are computed efficiently for a broad class of process models. Even if a process model does not satisfy the criteria for efficient computation of behavioural profiles, the effort for the profile computation has to be invested only once. Hence, this effort is independent of the number of cases for which conformance should be assessed, since the behavioural relations for a single case are derived directly. This is a major difference compared to the replay based conformance assessment. Replay of a *single* case may require exploration of a part of the system's state space.

Recently, the idea of conformance measurement based on behavioural relations was also picked up for the footprint that comprises the relations of the α-algorithm [446]. This approach exploits direct successorship of transitions and compares the footprint of a process model and a process log. Grounding in a complete process log, not single cases, is required as the footprint materialises only when the log meets certain completeness assumptions. As such, this approach illustrates that many of the concepts introduced in this chapter, e. g., those on root cause analysis, may also be transferred to conformance evaluation based on a different relational semantics.

9.2 BEHAVIOURAL PROFILES FOR CASES

This section introduces behavioural profiles for cases of a log. First, we formally discuss the notion of a case. In the context of net systems, a case is a non-empty observed sequence of transition occurrences. However, a case is not necessarily a valid firing sequences starting in the initial marking of the respective net system. A log comprises many of such cases.

Definition 9.2.1 (Case, Log)
Let T be a finite set of transitions. A *case* c over T is a finite sequence $c = \langle t_1, \ldots, t_n \rangle$, $n \in \mathbb{N}$, with $t_j \in T$ for all $1 \leqslant j \leqslant n$. A *log* over T is a finite set of cases $C = \{c_1, \ldots, c_m\}$, $m \in \mathbb{N}$, over T.

To define causal behavioural profiles for cases, we first have to clarify the notion of weak order for a case. Two transitions are in weak order in a case, if the first occurs before the second.

Definition 9.2.2 (Weak Order (Case))
Let $c = \langle t_1, \ldots, t_n \rangle$ be a case over T. A pair of transitions $(x, y) \in (T \times T)$ is in the *weak order relation* $\succ_c$, iff there exists two indices $j, k \in \mathbb{N}$, $1 \leqslant j < k \leqslant n$, for which holds $t_j = x$ and $t_k = y$.

Using this relation, we define the behavioural profile of a case.

Definition 9.2.3 (Behavioural Profile (Case))
Let $c = \langle t_1, \ldots, t_n \rangle$ be a case over T. A pair $(x, y) \in (T \times T)$ is in the following *profile relations*:

- The *strict order relation* $\rightsquigarrow_c$, iff $x \succ_c y$ and $y \not\succ_c x$.
- The *exclusiveness relation* $+_c$, iff $x \not\succ_c y$ and $y \not\succ_c x$.
- The *interleaving order relation* $\|_c$, iff $x \succ_c y$ and $y \succ_c x$.

$\mathcal{B}_c = \{\rightsquigarrow_c, +_c, \|_c\}$ is the *behavioural profile* of c.

As for the behavioural profile of a net system, a pair (x, y) is in *reverse strict order*, denoted by $x \rightsquigarrow_c^{-1} y$, if and only if $y \rightsquigarrow_c x$. We see that the properties discussed for a behavioural profile of a net system, cf., Section 4.1, hold also for the behavioural profile of a case. That is, the relations are mutually exclusive and, along with reverse strict order partition the Cartesian product of transitions of a case. In contrast to the profile of a net system, exclusiveness between two transitions can be observed in a case solely as a self-relation. For all pairs of transitions (x, y) for which we observe $x + y$ in a case, it holds $x = y$. For illustration purposes, consider the example case $c_2 = \langle I, A, C, B, G, F, O \rangle$. Here, it holds, e. g., $C \rightsquigarrow_{c_2} F$, $B \rightsquigarrow_{c_2}^{-1} A$, and $A +_{c_2} A$.

As the last step, we define the causal behavioural profile of a case. All transitions of a case are co-occurring.

Definition 9.2.4 (Causal Behavioural Profile (Case))
Let $c = \langle t_1, \ldots, t_n \rangle$ be a case over T and $\mathcal{B}_c = \{\rightsquigarrow_c, +_c, \|_c\}$ the behavioural profile of c.

- The *co-occurrence relation* $\gg_c = (T \times T)$ contains all pairs of transitions of the case.
- The set $\mathcal{B}_c^+ = \mathcal{B}_c \cup \{\gg_c\}$ is the *causal behavioural profile* of c.

9.3 CONFORMANCE MEASURES

This section introduces conformance measures based on behavioural profiles. First, we elaborate on the relation between the behavioural profile of a net system and the behavioural profile of a case. Second, we introduce measures for different conformance aspects. Third, we aggregate these measures to arrive at a single conformance value for a case. Finally, we discuss the influence of common noise types on our measures.

Type Subsumption of Profile Relations

There is a fundamental difference between a behavioural profile of a net system and of a case. The former defines relations based on the set of *all* possible firing sequences, whereas the latter considers only *one* observed sequence of transition occurrences. The relation between two transitions in a net system may be caused by different firing sequences. Hence, a different behavioural relation may be observed once only a single case is considered. For instance, transitions that may be enabled concurrently in a net system, C and F in our example in Figure 69, are related by interleaving order in the behavioural profile of the model, C||F. However, they may be related by strict order or reverse strict order in the profile of a case, even if the case represents a valid firing sequence. In our example case $c_2 = \langle I, A, C, B, G, F, O \rangle$, we observe $C \rightsquigarrow_{c_2} F$ as the behavioural relation for activities C and F.

To cope with this issue, we leverage the hierarchy between the relations of behavioural profiles (we neglect the co-occurrence relation at this stage) as discussed in Section 8.2. For the purpose of conformance analysis, we formalise this hierarchy between behavioural relations by a notion of *type subsumption*. Two behavioural relations are in this relation, if and only if the first relation is equal or weaker than the second. The following definition uses the notion of type equivalence of profile relations as introduced in Definition 6.2.2.

Definition 9.3.1 (Type Subsumption of Profile Relations)
Let $\mathcal{B}_1 = \{\rightsquigarrow_1, +_1, \|_1\}$ and $\mathcal{B}_2 = \{\rightsquigarrow_2, +_2, \|_2\}$ be two behavioural profiles. A relation $R_1 \in \mathcal{B}_1 \cup \{\rightsquigarrow_1^{-1}\}$ *subsumes the type* of a relation $R_2 \in \mathcal{B}_2 \cup \{\rightsquigarrow_2^{-1}\}$, denoted by $R_1 \sqsupseteq R_2$, iff either $R_1 \simeq R_2$, $R_1 \in \{\rightsquigarrow_1, \rightsquigarrow_1^{-1}\} \wedge R_2 = +_2$, or $R_1 = \|_1$.

We illustrate this concept using the net system in Figure 69 and case $c_2 = \langle I, A, C, B, G, F, O \rangle$. For transitions C and F, it holds C||F in the profile of the net system and $C \rightsquigarrow_{c_2} F$ in the profile of the case. The former specifies that C and F may occur in any order in a firing sequence, owing to the interleaving semantics of transitions that are enabled concurrently. The latter captures that the occurrences of C and F in the case are ordered. We see that both relations are in the type subsumption relation, $\| \sqsupseteq \rightsquigarrow_{c_2}$. Hence, we conclude on conformance regarding the profile relations for this particular pair of transitions.

Measures for Conformance Aspects

Our approach to conformance measurement separates two aspects. We assess whether the order of occurrence of transitions

in a case is valid based on the relations of the behavioural profile. The question of which transitions should occur in a case is addressed using the co-occurrence relation of the causal behavioural profile.

The measures that we will introduce for conformance measurement resemble those introduced in Section 7.2 for the quantification of consistency of an alignment between process models. Yet, they are different. First and foremost, they are based on the notion of type subsumption of profile relations instead of type equivalence. This is allows us to cope with the conceptual differences of behavioural profiles of process models and those of logs. Further, we separately measure the conformance based on the non-causal behavioural profile and on the co-occurrence relation, since both measures are normalised differently.

Behavioural profile conformance. The order of occurrences of transitions in a case should be in line with the behavioural relations of the respective net system. We analyse the Cartesian product of transitions in a case and determine whether the behavioural relation for a transition pair in the case is subsumed by the relation specified in the net system. This takes distinct transitions that are meant to be mutually exclusive into account. Those transitions are related by the exclusiveness relation of the behavioural profile of the net system. Once these transitions occur in a case, they are related by (reverse) strict order or interleaving order. Hence, the mutual occurrence defined by the net system is counted as being violated.

We define two degrees of behavioural profile conformance that differ in their normalisation. First, *model-relative* behavioural profile conformance is defined as the ratio of consistent behavioural relations relative to the number of transition pairings in the case. This degree considers all transition pairs that occur in the case. Hence, it directly depends on the number of transitions of the net system that have occurred already. Second, we neglect transition pairs that show interleaving order in the net system. Following on the argumentation on a hierarchy of profile relations, cf., Section 8.2, interleaving order can be interpreted as the absence of any order dependency. Interleaving order defined by the net system for two transitions cannot be violated by any case. We define *constraint-relative* behavioural profile conformance that is independent of the number of transitions in the case, but depends on the number of exclusiveness and strict order dependencies imposed by the net system.

Definition 9.3.2 (Behavioural Profile Conformance)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$ and $c = \langle t_1, \ldots, t_n \rangle$ a case over T_c with $T_c \subseteq T$. Let $\mathcal{B} = \{\rightsquigarrow, +, \|\}$ and $\mathcal{B}_c = \{\rightsquigarrow_c, +_c, \|_c\}$ be the behavioural profiles of S and c.

- The *set of profile consistent case pairs* $PC_c \subseteq (T_c \times T_c)$ contains all transition pairs (x, y), for which the profile relation in P subsumes the type of the relation in c, i. e., $\forall\, R \in (\mathcal{B} \cup \{\rightsquigarrow^{-1}\}), R' \in (\mathcal{B}_c \cup \{\rightsquigarrow_c^{-1}\})$ it holds $(xRy \wedge xR'y) \Rightarrow (R \sqsupseteq R')$.
- The degree of *model-relative behavioural profile conformance* of c to S is defined as

$$\mathcal{MBC}_c = \frac{|PC_c|}{|T_c|^2}.$$

- The degree of *constraint-relative behavioural profile conformance* of c to S is defined as

$$\mathcal{CBC}_c = \begin{cases} 1 & \text{if } \|_c = (T_c \times T_c), \\ \frac{|PC_c \setminus \|_c|}{|(T_c \times T_c) \setminus \|_c|} & \text{else.} \end{cases}$$

Both conformance degrees are between zero, no conformance at all, and one indicating full conformance. Computation of both degrees requires iteration over the Cartesian product of transitions in the case. Hence, the computation does not add to the complexity needed to compute behavioural profiles for net systems. Given case $c_2 = \langle I, A, C, B, G, F, O \rangle$ and the example in Figure 69, an order dependency imposed by the model is not satisfied. It holds $F \rightsquigarrow G$ according to the behavioural profile of the net system, whereas we have $F \rightsquigarrow_{c_2}^{-1} G$ in the profile of the case. In addition, the given case violates mutual occurrence of several transitions. The net system defines $B + C$, $B + F$, and $B + G$, whereas we have $B \rightsquigarrow_{c_2}^{-1} C$, $B \rightsquigarrow_{c_2}^{-1} F$, and $B \rightsquigarrow_{c_2}^{-1} G$ in the case.

Quantification of these violations relative to the number of considered transitions yields a degree of model-relative behavioural profile conformance of $\mathcal{MBC}_{c_2} = \frac{41}{49} \approx 0.84$ for this particular case. Once interleaving order is neglected, we derive a degree of constraint-relative behavioural profile conformance of $\mathcal{CBC}_{c_2} = \frac{38}{46} \approx 0.83$. Here, interleaving order between transitions C and F, and for B in relation to itself, is not considered in the conformance assessment.

Co-occurrence Conformance. Behavioural profile conformance focusses on the order of occurrence as captured by the behavioural profile. In Section 4.2, we discussed that the co-occurrence relation of the causal behavioural profile is largely independent of these relations. Hence, we consider co-occurrences of transitions by a separate conformance degree. Informally, we check whether all transitions for which the occurrence is implied by the current state of the case are in the case as well.

The ratio of the co-occurrence dependencies imposed by the net system that are satisfied by the case to all co-occurrence dependencies would be a straight-forward measure for this aspect. However, we want to consider also cases that may not

have completed yet. Transitions that are missing according to the co-occurrence relation may be added later on. Therefore, we consider only those transitions that are required to occur in the case by a co-occurrence dependency, for which we can deduce from the case that they should have already been observed. The latter is captured by the strict order relation. Consider case $c_4 = \langle I, C, E \rangle$ of our running example. According to the net system in Figure 69, any firing sequence that comprises transition I either also contains transitions A and O or leads to a marking in which both transitions are not dead. Transition A does not occur in the case, although we know that it should have been observed already owing to the strict order relation between A and both transitions, C and E. Transition O, in turn, is not required to occur in the case. The case does not contain any transition that is in strict order with O.

Again, there are two ways to normalise the conformance degree. First, the degree is normalised by the number of potential co-occurrence dependencies, i. e., the number of transitions that occur in the case or can be expected to occur in the case. We refer to this degree as *model-relative* co-occurrence conformance. Second, the degree is normalised based on the number of co-occurrence dependencies that are actually defined in the behavioural profile of the net system. We refer to this degree as *constraint-relative* co-occurrence conformance.

Definition 9.3.3 (Co-occurrence Conformance)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$ and $c = \langle t_1, \ldots, t_n \rangle$ a case over T_c with $T_c \subseteq T$. Let $\mathcal{B}^+ = \{\rightsquigarrow, +, \|, \gg\}$ and $\mathcal{B}^+_c = \{\rightsquigarrow_c, +_c, \|_c, \gg_c\}$ be the causal behavioural profiles of S and c.

- The *set of expected case transitions* $ET_c \subseteq T$ contains all transitions that occur in the case or that can be expected to occur in the case, i. e., $ET_c = T_c \cup \{t \in T \mid \exists\, t_1, t_2 \in T_c \,[\, t \rightsquigarrow t_2 \wedge t_1 \gg t \wedge (t_1 = t_2 \vee t_1 \rightsquigarrow t_2)\,]\}$.
- The *set of expected case transition pairs* $EP_c \subseteq (T \times T)$ is defined as $EP_c = (ET_c \times T) \setminus id_T$.
- The degree of *model-relative co-occurrence conformance* of c to S is defined as
$$\mathcal{MCC}_c = \frac{|(ET_c \times T_c) \setminus id_{T_c} \cap \gg| + |EP_c \setminus \gg|}{|EP_c|}.$$
- The degree of *constraint-relative co-occurrence conformance* of c to S is defined as
$$\mathcal{CCC}_c = \begin{cases} 1 & \text{if } \gg = \emptyset, \\ \frac{|(ET_c \times T_c) \setminus id_{T_c} \cap \gg|}{|EP_c \cap \gg|} & \text{else.} \end{cases}$$

Computation of both degrees requires determining the set of expected case transitions. These transitions are identified by ana-

lysing relations between three transitions, such that this step requires at most iteration over all transition triples of the net system. This takes cubic time with respect to the size of the net system. Once this set is determined, the degrees are derived by iterating at most over the Cartesian product of transitions of the net system. We conclude that the computation of these degrees does not add to the complexity needed to derive behavioural profiles for net systems.

We illustrate co-occurrence conformance using our running example and case $c_2 = \langle I, A, C, B, G, F, O \rangle$. Here, the co-occurrence $C \gg D$ is not satisfied. This is penalised as the case contains G and it holds $D \rightsquigarrow G$ in the net system. In other words, the occurrence of G in the case provides us with evidence that we should have observed D, too. The same holds true for transition E, which can be expected to be in the case. Computation of the model-relative degree of co-occurrence conformance yields a value of $\mathcal{MCC}_{c_2} = \frac{64}{72} \approx 0.89$. Eight violations are considered, relative to the number of potential co-occurrence dependencies. As an example, the transition pair (C, A) is considered for normalisation, although it holds $C \not\gg A$. Computation of the constraint-relative degree of co-occurrence conformance yields a value of $\mathcal{CCC}_{c_2} = \frac{36}{44} \approx 0.82$. 36 out of 44 co-occurrence dependencies of transitions that can be expected to be in the case are satisfied. For case $c_4 = \langle I, C, E \rangle$, the absence of transition A is penalised. Strict order between transitions A and B, $A \rightsquigarrow C$, indicates that transition A should have been observed already in the case. For the example case c_4, we compute conformance values of $\mathcal{MCC}_{c_4} = \frac{9}{12} = 0.75$ and $\mathcal{CCC}_{c_4} = \frac{5}{8} \approx 0.63$.

Note that the co-occurrence conformance degree may be *overestimated*. An example for this phenomenon would be the case $\langle I, A, J \rangle$ for the net system in Figure 69. This net system defines a co-occurrence dependency $J \gg B$. In our analysis of co-occurrence conformance, the absence of transition B in the case $\langle I, A, J \rangle$ would not be penalised. There is no transition in the case that is in strict order with B and, therefore, would provide us with sufficient evidence that transition B should have already been observed.

Aggregated Conformance Measures

The degrees for conformance aspects introduced earlier are the foundation for aggregated conformance measures for a case. An aggregated measure is defined as the sum of the enumerators divided by the sum of the denominators of the respective degrees. Hence, differences in the denominators are taken into account. Those differences stem from transitions that do not occur in the case but are expected to occur. These transitions are considered

in the co-occurrence conformance measures, but not in the behavioural profile conformance measures. We first combine the two model-relative measures.

Definition 9.3.4 (Model-Relative Case Conformance)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$ and $c = \langle t_1, \ldots, t_n \rangle$ a case over T_c with $T_c \subseteq T$. Let $\mathcal{B}^+ = \{\rightsquigarrow, +, \|, \gg\}$ and $\mathcal{B}^+_c = \{\rightsquigarrow_c, +_c, \|_c, \gg_c\}$ be the causal behavioural profiles of S and c, PC_c the set of profile consistent case pairs, ET_c the set of expected case transitions, and EP_c the set of expected case transition pairs. The *model-relative case conformance* of c to S is defined as

$$\mathcal{MC}_c = \frac{|PC_c| \; + \; |(ET_c \times T_c) \setminus id_{T_c} \cap \; \gg| + |EP_c \setminus \; \gg|}{|T_c|^2 \; + \; |EP_c|}.$$

Constraint-relative case conformance builds upon the constraint-relative measures for behavioural profile conformance and co-occurrence conformance.

Definition 9.3.5 (Constraint-Relative Case Conformance)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$ and $c = \langle t_1, \ldots, t_n \rangle$ a case over T_c with $T_c \subseteq T$. Let $\mathcal{B}^+ = \{\rightsquigarrow, +, \|, \gg\}$ and $\mathcal{B}^+_c = \{\rightsquigarrow_c, +_c, \|_c, \gg_c\}$ be the causal behavioural profiles of S and c, PC_c the set of profile consistent case pairs, ET_c the set of expected case transitions, and EP_c the set of expected case transition pairs. The *constraint-relative case conformance* of c to S is defined as

$$\mathcal{CC}_c = \begin{cases} 1 & \text{if } \gg = \emptyset \text{ and} \\ & \| = (T_c \times T_c), \\ \frac{|PC_c \setminus \|_c| + |(ET_c \times T_c) \setminus id_{T_c} \cap \gg|}{|(T_c \times T_c) \setminus \|_c| + |EP_c \cap \gg|} & \text{else.} \end{cases}$$

Table 6 illustrates the results for all conformance measures for the net system and the five exemplary cases introduced in Section 9.1. As expected, for the first case c_1, which represents a valid firing sequence of the net system, all measures indicate full conformance. In contrast, the dependencies defined by the relations of the behavioural profile of the net system are not fully satisfied in the second case c_2. For instance, the exclusiveness in the model between transitions B and C is broken in the case. Co-occurrences, e. g., between transitions C and D, are not completely respected either. For case c_3, we make similar observations leading to overall conformance values between 0.74 and 0.85. Here, the difference in the normalisation of our two aggregated conformance measures becomes visible. An assessment that is relative to the number of transitions leads to a higher conformance value. Pairs of transitions without explicit behavioural dependency according to the profile of the net system lower the

Table 6: Conformance results for the example cases, see Section 9.1.

$\mathcal{CBC}$ Constr.-Rel. Beh. Profile	$\mathcal{MBC}$ Model-Rel. Beh. Profile	$\mathcal{CCC}$ Constr.-Rel. Co-occur.	$\mathcal{MCC}$ Model-Rel. Co-occur.	$\mathcal{CC}$ Constr.-Rel. Conf.	$\mathcal{MC}$ Model-Rel. Conf.
$c_1 = \langle I, A, E, C, D, F, G, O \rangle$					
1.00	1.00	1.00	1.00	1.00	1.00
$c_2 = \langle I, A, C, B, G, F, O \rangle$					
0.83	0.84	0.82	0.89	0.82	0.87
$c_3 = \langle I, A, B, J, H, B, O, G \rangle$					
0.80	0.84	0.69	0.85	0.74	0.85
$c_4 = \langle I, C, E \rangle$					
1.00	1.00	0.63	0.75	0.80	0.86
$c_5 = \langle F, C, D, G \rangle$					
1.00	1.00	0.50	0.62	0.64	0.72

influence of transition pairs for which we detect a violation. Regarding case c_4, we discussed that there are transitions missing in the case. This deviation impacts on the conformance degrees that are based on the co-occurrence relation. The degrees based on the behavioural profile equal one, as case c_4 does not violate any order or exclusiveness dependencies. Similarly, case c_5 shows full behavioural profile conformance since the order of occurrence of transitions is in line with the net system. Still, case c_5 is incomplete. It represents a subsequence of firing sequence of the net system, such that the first part of the firing sequence is missing, i. e., transitions I, A, and E. Consequently, various co-occurrence dependencies are violated.

The conformance measures based on the relations of the behavioural profile, $\mathcal{CBC}$ and $\mathcal{MBC}$, are largely independent of those that are grounded on the co-occurrence relation, $\mathcal{CCC}$ and $\mathcal{MCC}$. This follows from the fact that the respective behavioural relations are largely orthogonal, cf., Property 4.2.2 in Section 4.2. Hence, the aggregated measures, $\mathcal{CC}$ and $\mathcal{MC}$, shall be used to take the complete spectrum of behavioural relations of the causal behavioural profile into account.

Conclusions based on the actual conformance values can only be drawn against the background of a concrete business process and environment. There is a variety of factors that influence the question whether a certain degree of non-conformance is acceptable. The severity of the implications that follow from non-conforming behaviour and the reliability of the logging mechanism would be examples for such factors.

Given the conformance values for single cases, conformance of a complete log is evaluated based on the average conformance values of all cases. The arithmetic mean of the conformance values along with its standard deviation indicate to which the degree the cases deviate from the model and how these deviations are distributed among the cases.

The Influence of Noise

Conformance measurement assumes that a normative process model exists and that the log gives an accurate account of how individual cases have been processed. Research on process mining has acknowledged the existence of noise in real-world logs and implemented measures to deal with it. Noise stems among others from inaccurate logging mechanisms in information systems or race conditions when writing two log entries. Also, the execution order of activities may not be enforced explicitly or people may deliberately work around the system.

In the following paragraphs, we discuss the influence of noise on the conformance measures introduced in the previous section. This overview helps to pin down the computed conformance values with respect to the noise that can be expected in a certain setting. We discuss noise patterns along the classification introduced in [510, 511], which has been extended in [174]. There are two major categories of noise: *missing parts of cases* and *perturbation*.

Missing parts of a case are caused by a logging mechanism of a process-aware information system that was not available for a particular period of time. A missing head suggests that recording started only after the case was already in processing; a missing trail can result from cases that are still processed when the analysis period is closed; and a missing episode may stem from a temporarily deactivated logging mechanism, see Figure 70. These three patterns have in common that a part of the original case is missing. This category of noise does not influence the order between transition occurrences in the case. Therefore, the behavioural profile conformance measures are not affected by this kind of noise. The co-occurrence measures, in turn, penalise the missing parts. This penalty depends on how many co-occurrence dependencies exist between the transitions of the missing part and the remainder of the case.

Perturbation involves a wrong recording of transition order, the wrong recording of an additional event, or the recording of alien events. These perturbations affect the conformance measures to a different degree. Consider the pattern that two events are recorded in wrong order, for instance due to a race condition in the logging mechanism, Figure 71, top. If there is an or-

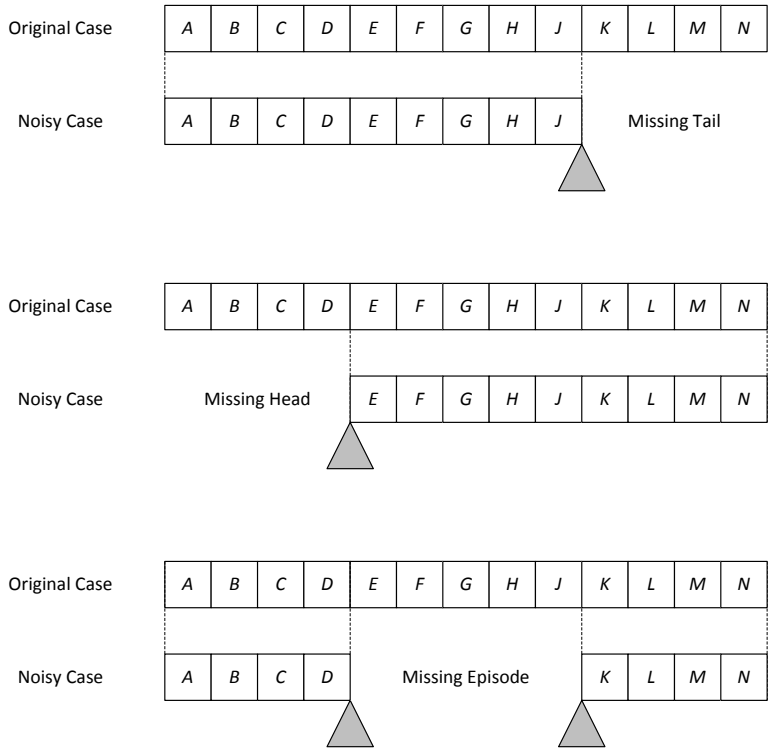


Figure 70: Types of noise with missing parts of a case, see also [174].

der dependency between these two transitions, we now observe a violation for the perturbed case. If both were in interleaving order anyway, then the conformance degree is not affected. In the same vein, order dependencies between the two perturbed transitions and the transitions that occur between them are affected. We observe violation, if a transition was not in interleaving order, but in strict order with one of the perturbed transitions. Co-occurrence dependencies are not violated by this kind of noise, since the set of transitions in the case does not change. Figure 71, middle, shows the pattern of an additional recording of an event. Here, the perturbed case shows violations for all transitions between the first and the second record, if an order dependency was defined between them and the repeated transition. Again, co-occurrences are not violated. If an alien event is recorded, Figure 71, bottom, neither order relations nor co-occurrence relations are violated.

To conclude, behavioural profile conformance turned out to be robust against noise with missing parts of cases and alien events. Co-occurrence conformance is robust against perturbation noise.

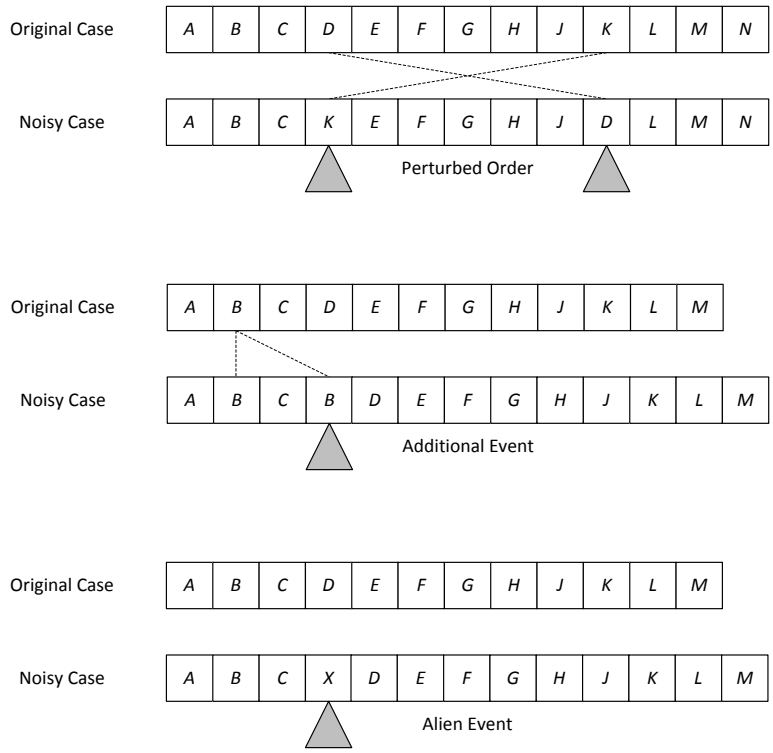


Figure 71: Types of noise with perturbation, see also [174].

9.4 DIAGNOSTICS

The measures introduced in Section 9.3 give an insight into the conformance of a single case. To identify reasons for non-conforming processing, diagnostic information on conformance violations has to be derived. In particular, the root cause of a conformance violation should be isolated. In this section, we first discuss concepts for root cause analysis on the level of a single case. Then, we turn the focus on diagnostic information for a log.

Root Cause Analysis for a Single Case

Non-conforming processing of a case causes violations of behavioural dependencies. Those are given as a set of triples, referred to as *violation*, each consisting of a pair of transitions along with the violated relation of the causal behavioural profile. A set of violations may be traced back to a few transitions. Such transitions have to be identified to highlight transition occurrences that are most problematic as they heavily affect the overall conformance assessment. The ratio between the violated dependencies that relate to a transition to all violated dependencies is referred to as the *violation impact* of the transition. This value measures how

Table 7: Feedback on non-conformance for case c_3 of the example, see Section 9.1.

Case $c_3 = \langle I, A, B, J, H, B, O, G \rangle$	
Violations	(F,C,≫),(C,F,≫),(G,D,≫),(F,E,≫), (G,E,≫),(D,F,≫), (F,D,≫),(E,C,≫),(E,D,≫),(C,E,≫),(G,F,≫),(E,F,≫), (D,C,≫),(G,C,≫),(D,E,≫),(C,D,≫),(H,G,+),(J,G,+), (B,G,+),(G,B,+),(G,J,+),(G,H,+),(O,G,$\rightsquigarrow^{-1}$),(G,O,$\rightsquigarrow$)
Violation Impact	$\mathcal{VI}(G) = 0.5$ $\mathcal{VI}(D) = \mathcal{VI}(E) = \mathcal{VI}(C) = \mathcal{VI}(F) = 0.29$ $\mathcal{VI}(B) = \mathcal{VI}(H) = \mathcal{VI}(J) = \mathcal{VI}(O) = 0.08$

much of the observed non-conforming behaviour is related to the respective transition.

Definition 9.4.1 (Violation and Violation Impact)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$ and $c = \langle t_1, \ldots, t_n \rangle$ a case over T_c with $T_c \subseteq T$. Let $\mathcal{B}^+ = \{\rightsquigarrow, +, \|, \gg\}$ and $\mathcal{B}^+_c = \{\rightsquigarrow_c, +_c, \|_c, \gg_c\}$ be the causal behavioural profiles of S and c, PC_c the set of profile consistent case pairs, and ET_c the set of expected case transitions.

- The set of *violation* $\mathcal{V}_c \subseteq (T \times T \times \mathcal{B}^+)$ for c contains all pairs (x, y, R) such that $(x\ R\ y)$ and either $(x, y) \notin PC_c$ or $(x \in ET_c \wedge x \gg_c y \wedge x \neq y \wedge y \notin T_c)$.
- The *violation impact* of a transition $t \in T$ is defined as

$$\mathcal{VI}(t) = \frac{|\{(x, y, R) \in \mathcal{V}_c \mid (x = t) \vee (y = t)\}|}{|\mathcal{V}_c|}.$$

Table 6 showed that four out of five of the example cases have overall conformance values below one. Taking case c_3 as an example, Table 7 illustrates the respective violations. Those indicate the concrete problems that have been found for the case. Table 7 also illustrates the violation impact of single transitions. The occurrence of transition G turns out to be most problematic for case c_3. One-half of the violations relate to this transition. Its occurrence can be seen as the root cause for the non-conformance of this particular case.

Root Cause Analysis for a Log

When investigating conformance of a log, feedback on violations should not be limited to single cases. Instead, the frequency with which a violation is observed has to be known to identify the reasons for non-conforming processing in general. Dependen-

Table 8: Violations (support > 1) for the example log, see Section 9.1.

Violations	Support
(C,E,≫),(D,E,≫),(G,E,≫),(F,E,≫)	3
(G,D,≫),(B,G,+),(E,D,≫),(C,D,≫),(I,A,≫), (E,A,≫),(F,D,≫),(G,B,+),(C,A,≫)	2

cies between violations are valuable feedback as well. A certain violation may be caused by a violation that happened before.

We address the need for aggregated analytic information on non-conformance using the notion of a violation as introduced for a single case. Our analysis adapts the notions of *support* and *confidence* known from the field of association rules mining [10, 9]. Association rules mining identifies patterns that are built of items given a set of transactions. These transactions, in turn, are built of items. Adapted to our setting, a transaction is represented by a case and an item is a violation, which may be observed in a case. We define support for a violation as the number of cases in a log, in which the violation is observed.

Definition 9.4.2 (Support for Violation)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$ and $C = \{c_1, \ldots, c_n\}$ a log over T_C with $T_C \subseteq T$. Let $\mathcal{B}^+ = \{\rightsquigarrow, +, \|, \gg\}$ be the causal behavioural profiles of S.

- The set of logs supporting a violation $v \in (T \times T \times \mathcal{B}^+)$ in C is defined as $\mathcal{SU}(v) = \{c_i \in C \mid v \in \mathcal{V}_{c_i}\}$.
- The *support* for a violation $v \in (T \times T \times \mathcal{B}^+)$ in C is defined as $\mathrm{sup}(v) = |\mathcal{SU}(v)|$.

Table 8 shows the violations for the cases of the example log, see Section 9.1, which have a support larger than one. It illustrates that there are four violations that are observed in three out of five cases. All of them represent violations of co-occurrences that imply the execution of transition E. This observation provides a starting point for the analysis of the respective process. The reasons for missing occurrences of transition E have to be determined, as those are the root causes for non-conforming processing.

The analysis of support for violations helps to separate frequent and rare violations. Analysis of non-conformance is even more effective if implications between violations are taken into account. A deviation from the processing as specified in the net system may cause several subsequent violations. Detection of these implications allows us to focus on the actual cause of a series of violations. To address this demand, we adopt the notion of confidence of association rules. Confidence relates rules

between items to their statistical significance. Thus, it reflects the strength of a rule. In our setting, a rule is an implication between two violations.

Definition 9.4.3 (Confidence for Violation Rules)
Let $S = (N, M_0)$ be a net system with $N = (P, T, F)$ and $C = \{c_1, \ldots, c_n\}$ a log over T_C with $T_C \subseteq T$. Let $\mathcal{B}^+ = \{\rightsquigarrow, +, \|, \gg\}$ be the causal behavioural profiles of S. For two distinct violations $v_1, v_2 \in (T \times T \times \mathcal{B}^+)$ the *confidence* for a violation rule from v_1 to v_2 is defined as

$$conf(v_1 \Rightarrow v_2) = \begin{cases} 0 & \text{if } sup(v_1) = 0, \\ \frac{|\mathcal{SU}(v_1) \cup \mathcal{SU}(v_2)|}{sup(v_1)} & \text{else.} \end{cases}$$

Analysis of violation rules is reasonable solely for violations for which the support exceeds a certain threshold in the log. Similarly, only rules that exceed a certain threshold with respect to their confidence shall be investigated.

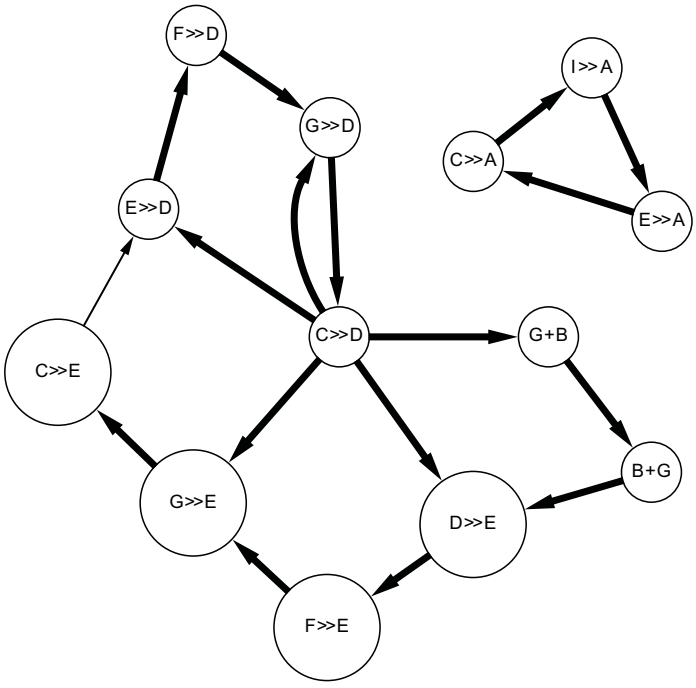


Figure 72: Rules between violations (confidence > 0.6) for the example log, see Section 9.1.

Figure 72 depicts the rules between violations for the example log introduced in Section 9.1. Nodes depict violations that show a support larger than one, which have also been listed in Table 8. The node size reflects the different support values. Edges represent rules between violations for which the confidence value is above the threshold of 0.6. The edge strength depends on the confidence value. In our example, all except one rule show a

confidence value of one. That is, the occurrence of the source violation implies the occurrence of the target violation. Note that we did a transitive reduction for the edges in the graph. A transitive reduction is not unique in case of a cyclic graph and identification of the minimal transitive reduction is an NP-complete problem [390]. To provide an overview of the interplay of violations, however, the identification of one non-minimal reduction is sufficient.

Figure 72 illustrates that there are two clusters of violations that manifest in disconnected subgraphs. These clusters represent violations that occur independent of each other and, therefore, may be analysed separately. Focussing on the bigger subgraph, the violation related to the co-occurrence between transitions C and D implies various other violations. Although this violation cannot be seen as the only root cause – it is part of a cycle of violation rules – there is some evidence that this violation is fundamental and many other violations are causally dependent. Hence, the implementation of the process should be investigated for reasons that break the co-occurrence between transitions C and D.

We restricted our discussion on rules between two violations. However, the introduced concepts may be lifted to rules between more than two violations in a straight-forward manner.

9.5 EXPERIMENTAL EVALUATION

To demonstrate and evaluate our approach to conformance analysis, we implemented all introduced concepts in a prototypical tool and applied them in a case study on the Security Incident Management Process (SIMP). In this section, first, we give background information on this process. Second, we present analysis results for a log using the proposed conformance measures. Third, we apply the concepts introduced for root cause analysis of non-conformance to the log.

SIMP Background

The Security Incident Management Process (SIMP) is an issue management process operated by IBM's global service delivery centres. We focus on the SIMP as it is run in a service delivery centre that provides infrastructure management and technical support to customers. The process and the log have been minimally modified to remove confidential information. Figure 73 shows the SIMP as a net system solicited from domain experts.

When a customer reports a problem or requests a change, an issue is created, spawning a new instance of the process. Details about the issue may be updated, a plan to resolve the issue

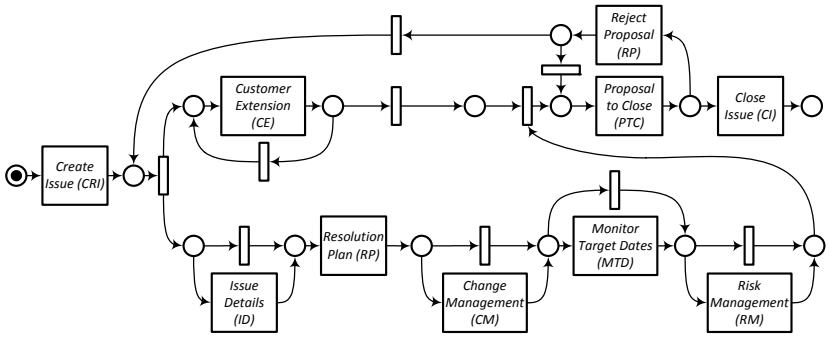


Figure 73: Net system of the Security Incident Management Process (SIMP).

must be created, and change management related activities may be performed if required. Then, target dates for issue resolution may be monitored and relevant risks may be documented. Concurrently, a customer extension of target dates may be processed. Once the steps for resolution are taken and verified, the resolver must propose to close the issue. Based on the evidence that the issue is indeed resolved, the issue creator may close the issue. Otherwise, the proposal must be rejected.

For the SIMP, we analysed 852 cases each consisting of a set of log entries. Such a log entry has a transition name, a description, and the time-stamp marking the time of occurrence of the transition. Although the process is standardised and documented, it is not orchestrated via workflow tools in the IBM's global service delivery centre under investigation. Instead, it is manually carried out. Hence, employees are free to deviate from the process. As a result, the cases may or may not specify valid firing sequences of the net system. The log has been created using a proprietary tool, in which an employee submits the execution of a certain activity, i. e., the occurrence of a certain transition. Correlation of log entries to cases has been managed explicitly by the logging tool.

Conformance Measures

For each case, we analysed its conformance using the introduced measures. Table 9 provides a summary of this analysis. It shows the average conformance value of all cases (using the arithmetic mean) along with the standard deviation, the observed minimal and maximal conformance values, and the share of fully conforming cases (all conformance measures yield a value of one). The conformance values were discussed with the manager of the process. The average values reflect the manager's perception that SIMP is running satisfactory and most cases are handled

Table 9: Conformance results for the SIMP derived from 852 cases.

	$\mathcal{CBC}$ Constr.-Rel. Beh. Profile	$\mathcal{MBC}$ Model-Rel. Beh. Profile	$\mathcal{CCC}$ Constr.-Rel. Co-occur.	$\mathcal{MCC}$ Model-Rel. Co-occur.	$\mathcal{CC}$ Constr.-Rel. Conf.	$\mathcal{MC}$ Model-Rel. Conf.
Avg	0.98	0.99	0.96	0.96	0.97	0.97
StDev	0.06	0.04	0.11	0.09	0.08	0.069
Min	0.08	0.31	0.60	0.60	0.53	0.58
Max	1.00	1.00	1.00	1.00	1.00	1.00
Share of conforming Cases						
	78.64%	78.64%	84.39%	84.39%	76.29%	76.29%

in a conforming way. Although up to a quarter of the cases are not in line with the model, the high average values and the low standard deviation for our conformance measures indicate that there are solely marginal deviations in most cases . Still, as the minimum values show, it was also possible to identify cases of low conformance. Behavioural profile conformance values of 0.08 or 0.31 represent outliers that have been caused by a case that mixed the log entries for two separate process instances. Moreover, the different normalisations of conformance degrees, either based on the number of dependencies or the number of considered transitions, does not affect the conformance measurement for our case study significantly.

We are not able to directly compare our conformance results with the fitness measure proposed in [107] and discussed in Section 9.1. This is mainly due to the inherent complexity of the partial state space exploration. Even a maximally reduced net system of the SIMP contains a lot of silent transitions caused by activities that may be skipped. This leads to a significant increase of the size of the state space that must be investigated when trying to replay a case. As a consequence, fitness computation with the fitness plugin in ProM [463] (version 5), was possible for all cases only with the most greedy strategy. However, these results are of a limited validity. They highly underestimate the conformance values. Further, computation of the compliance values for the process log took around 15 seconds, whereas computation of the proposed conformance measures was done within milliseconds.

Root Cause Analysis

For all SIMP cases, we collected all observed violations. Figure 74 illustrates the amount of the collected violations and how

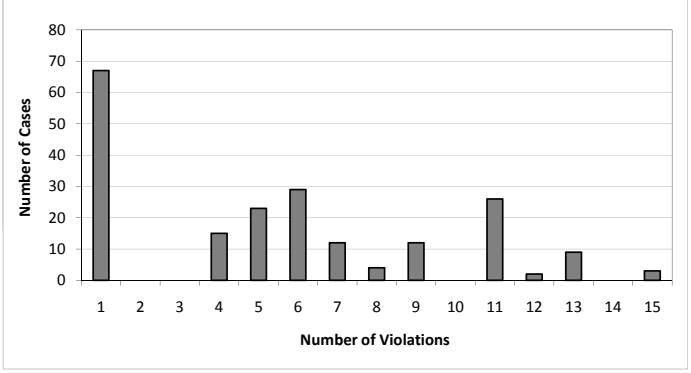


Figure 74: Number of SIMP cases (out of 852) relative to the number of observed violations.

Table 10: Feedback on non-conformance for a dedicated SIMP case.

Violations	(CLI,CLI,+),(CM,CE,≫),(CRI,CE,≫), (RPTC,CE,≫),(CLI,CE,≫),(PTC,CE,≫),(RP,CE,≫)
Violation Impact	$\mathcal{VI}(\text{CE}) = 0.86$ $\mathcal{VI}(\text{CLI}) = 0.29$ $\mathcal{VI}(\text{RP}) = \mathcal{VI}(\text{CM}) = 0.14$ $\mathcal{VI}(\text{CRI}) = \mathcal{VI}(\text{RPTC}) = \mathcal{VI}(\text{PTC}) = 0.14$

they are distributed over the cases. For each number of violations, the chart depicts the number of SIMP cases that showed an according number of violations. For nearly 70 cases we observe a single violation. At most 15 violations are detected in a single case. We conclude that in our setting, the set of violations for a single case can still be handled by a process analyst.

Table 10 illustrates the results of the root cause analysis for a single case. It depicts the violations for a dedicated case along with the violation impact of the respective transitions, see Figure 73 for the resolution of the transition abbreviations. The chosen case has an overall conformance value of 0.86 or 0.91, depending on the applied normalisation. Hence, it is a typical example for a non-conforming case that shows minor behavioural deviations from the behaviour as defined by the net system. For this case, there is a single transition that participates in more than 80% of the violations. The transition representing the *customer extension* (*CE*) can, therefore, be seen as the root cause of non-conformance of this case. The violated dependencies all require the occurrence of transition *CE*. Hence, the absence of this transition causes most of the violations. Apart from the dependencies related to transition *CE*, exclusiveness as a self-relation for

Table 11: Violations (support > 20) observed in the set of 852 SIMP cases.

Violations	Support
(CLI,CLI,+)	177
(CRI,RP,≫),(CE,RP,≫),(CLI,RP,≫),(PTC,RP,≫)	109
(CRI,CE,≫),(CLI,CE,≫),(PTC,CE,≫),(RP,CE,≫)	74
(CM,CE,+)	35
(CM,RP,+)	34

transition *close issue* (*CLI*) is violated. The case comprises two log entries that report an occurrence of this transition, whereas the net system allows for at most one occurrence.

We discussed the root cause analysis just for one exemplary case. However, a review of our results suggests that similar observations can be made for most of the non-conforming cases.

Turning the focus on the root cause analysis for the whole log, Table 11 lists the violations with the highest support in the set of 852 SIMP cases. The most frequently observed violation relates to the transition *close issue* (*CLI*). 177 cases record at least two occurrences of this transition, which is not in line with the net system, see Figure 73. In a large number of cases, co-occurrence dependencies that require the presence of the transition representing *resolution plan* (*RP*) or *customer extension* (*CE*) are violated. Although both transitions must occur to complete a case, they are absent in a large number of cases.

With these results, the necessity to deviate from the standard processing as illustrated can be evaluated by the management of the Security Incident Management Process. Any judgement on whether these deviations are acceptable can only be done once the reasons for deviation have been investigated in the concrete cases. Such investigations have not yet been complete for SIMP as illustrated in this case study.

Finally, we visualise the identified implications between violations in Figure 75. In this graph, nodes depict violations with a support larger than 20 in the collection of 852 cases. Edges represent rules between them that have a confidence above the threshold of 0.6. The node size reflects the support values and the edge strength correlates with the confidence value. The graph suggests that the violation related to the self-relation of transition *close issue* (*CLI*) is independent of the other frequently observed violations.

Violations of the co-occurrence dependencies involving transitions *RP* and *CE* build clusters of rules of high confidence. Hence, these violations always occur together in a case. Still, both clusters

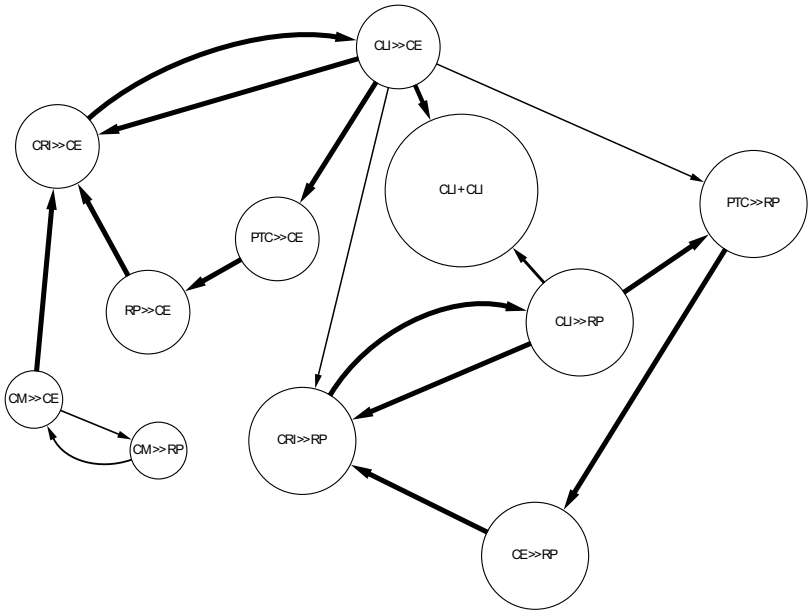


Figure 75: Rules between violations (confidence > 0.6) observed in the set of 852 SIMP cases.

are rather independent of each other. We conclude that the absence of transitions *RP* and *CE*, along with the repeated occurrence of transition *CLI*, have to be seen as independent root causes of non-conforming behaviour in the SIMP log.

9.6 RELATED WORK

Conformance measurement is at the core of process mining [11, 99, 457, 460, 446], which we discussed already in Section 8.6.

In Section 9.1, we mentioned various approaches for measuring conformance of a process log by using a replay method [183, 511, 395, 107, 176]. Some approaches quantify conformance as the share of cases of a log that can be replayed in the process model [183, 511]. Other approaches replay a case step-wise in the process model and quantify the number of valid execution steps [395, 107, 176].

Replay based conformance analysis is continuously improved. Recent work formulates the question of which is the *optimal* replay of a case as a search problem [6]. The latter is then solved using standard search algorithms. Another trend is to increase flexibility of conformance measures. In particular, activities may be distinguished regarding their importance [7]. By assigning costs to different deviations from the process model, conformance measures may be customised towards a certain setting. For instance, skipping a reporting step in a process may be considered to be less crucial than skipping a payment activity. Our

approach neglects such differences. However, similar extensions can be realised by assigning costs to certain behavioural relations (or to activities and, therefore, to all relations that relate to these activities). This cost would then be considered as a weighting factor when counting the violated behavioural relations.

Our concepts for root cause analysis relate to techniques for the visualisation of log data to enable effective analysis. To this end, dotted chart analysis to assess the performance of business operations with a focus on their time dependencies has been advocated in [428]. Such a chart supports the manual analysis of long-running instances. Another approach leverages multi sequence alignment techniques known from bioinformatics to construct a trace alignment [59]. Once an alignment between cases has been established, patterns of common behaviour and rare deviations may be identified. The drawback of this approach is its complexity. Finding an optimal alignment for a set of cases is a computationally hard problem. Nevertheless, [59] already showed the application of the technique in two case studies. An alignment assumes a different perspective compared to our feedback on violations of behavioural dependencies. Hence, both approaches can be seen as complementary.

The conformance of process execution with normative process models is also an important aspect of role-based access control (RBAC). In essence, role-based access control deals with the specification and enforcement of constraints that relate to order and exclusiveness of roles or subjects executing particular activities. Such constraints include, among others, separation of duty requirements. Separation of duty implies that either particular activities have to be exclusive altogether, or that those roles or subjects executing a pair of activities have to be exclusive (also referred to as four-eye principle) [276, 12, 160]. The major share of research in this area has focussed on the specification and verification of RBAC policies [48, 159, 338], among others on consistency and satisfiability of constraint sets [436, 94], and on engineering and enforcement by design [434]. Log files have been used for mining roles in this area [250, 326], while an a posteriori compliance control has only been considered recently [152, 170, 5]. Our concepts inform this stream of research. Once process models are annotated with RBAC constraints as defined in [521] and a log includes role and subject information, the relations of the causal behavioural profile can be leveraged to check separation of duty constraints.

9.7 CONCLUSION

This chapter introduced an approach to conformance measurement based on behavioural profiles. We defined behavioural pro-

files for cases of a process log and proposed a set of conformance measures. In essence, these measures quantify to what degree the behavioural relations as induced by the process model are respected by the log. We discussed alternatives for normalising our measures and also elaborated on the influence of common noise patterns on them. Then, we focussed on concepts that enable effective root cause analysis of non-conformance. Diagnostic information is provided on the level of single cases, and for a complete process log. We tested all concepts in a case study with an international service provider.

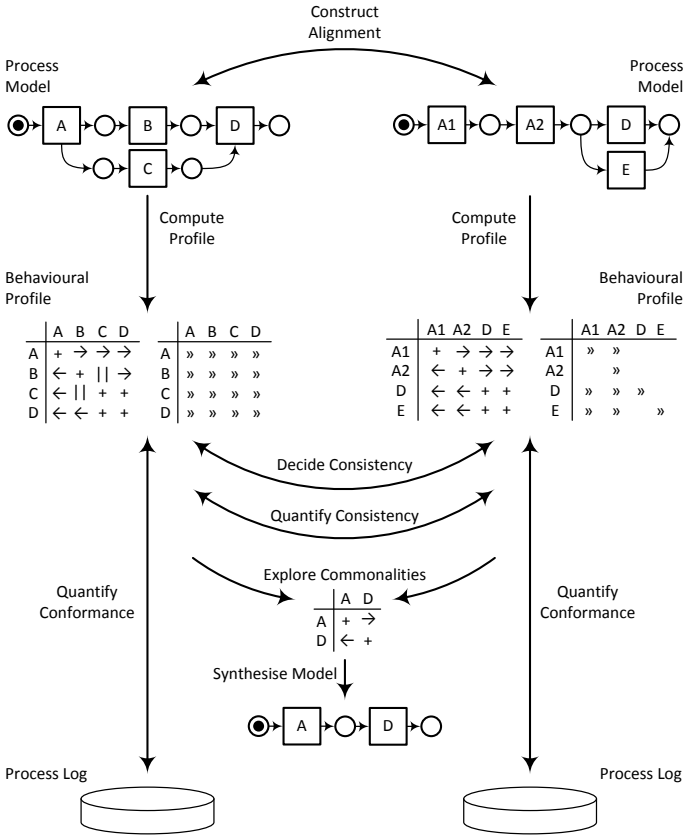
Compared to existing work, we take a fundamentally different approach to evaluate conformance of process logs. The application of behavioural profiles has several strengths. First, it allows us to give feedback on non-conformance directly in terms of violated behavioural dependencies. This feedback supports the interpretation of conformance values. For the SIMP of our case study, for instance, we showed that a great share of non-conformance stems from the repetition of the activity to close the issue. Second, our approach largely avoids computation intensive operations. Only the derivation of the behavioural profile for the process model may require exponential time to the size of the process model, if the assumptions for the efficient computation techniques are not met. Still, this effort is required only once and not for every single case.

As for the consistency analysis of alignments between process models, the efficiency of our approach is traded for accuracy. Our approach is grounded on a behavioural abstraction that neglects certain behavioural aspects. Only deviations that materialise in the behavioural relations can be detected and influence the conformance measurement. Methods that are based on a replay of cases, therefore, are sensitive also to non-conformance that cannot be discovered by our approach. The level of accuracy of our approach suffices to draw conclusion on how consistency results obtained between process models relate to the observed behaviour, as all measures have a unified formal grounding. For other use cases of conformance analysis, however, the abstracted aspects may turn out to be important. If so, there is potential to combine the presented approach with replay based conformance analysis. On the one hand, the efficiency of our approach allows us to apply it first to isolate cases that show a large non-conformance. Then, replay based methods are applied only for a subset of cases. On the other hand, the feedback given by the different approaches on non-conformance may be combined to increase its usefulness for process analysts.

In this chapter, we introduced the techniques for root cause analysis for the use case of conformance measurement. However, some of these concepts are also useful for consistency ana-

lysis of alignments. Consider, for instance, the presented notion of violation impact of a transition. It measures how much of the observed non-conforming behaviour is related to the respective transition. It may be lifted to the setting of an alignment and the degrees of consistency as proposed in Section 7.2. Then, the inconsistency that is related to a single transition of an alignment can be quantified. As such, this information is useful for the extraction of a consistent sub-alignment discussed in Section 7.4 to support change propagation. To obtain a consistent sub-alignment, we step-wise remove transitions from an alignment according to their violation impact, starting with the highest value.

10 CONCLUSIONS



The research presented in this thesis centred on the question of to how to assess behaviour consistency of related process models. Here, related means that the process models refer to the same business process. To address this question, we came up with an approach that starts with the construction of an alignment and ends with a manifold analysis of behaviour consistency. To this end, we took a relational approach that builds on the notion of a behavioural profile of a process model.

In this chapter, we briefly summarise our contributions in Section 10.1. Then, we discuss the concept of a behavioural profile in the broader context. In Section 10.2, we review several use cases in which the formalism underlying our analysis of behaviour consistency has also been applied. Finally, we turn the focus on the limitations of our work. Section 10.3 discusses open issues regarding the analysis of behaviour consistency and outlines directions for future research.

10.1 SUMMARY OF THE RESULTS

This thesis presented a framework to assess behaviour consistency of related process models. In the following, we briefly summarise the main results proposed as part of this framework.

The ICoP Framework

The identification of complex correspondences between process models is a challenging problem that has not been addressed before. Taking up techniques known from the field of data integration and ontology matching, we presented the ICoP framework. It provides a modular and adaptable architecture for the implementation of matchers to detect complex 1:n correspondences between process models. We validated the framework experimentally and showed that the implemented matchers indeed discover complex correspondences. Even though the ICoP framework has certain limitations, it is a first step towards (semi-) automated support for process model matching that goes beyond elementary 1:1 correspondences.

Behavioural Profiles and their Computation

Our approach to the analysis of behaviour consistency centres on the concept of a behavioural profile. We introduced this concept formally for net systems, elaborated on its properties, and discussed commonalities and differences to various related behavioural concepts. Further, to accommodate for different requirements regarding behavioural analysis, we distinguished two variations of the concept, behavioural profiles and causal behavioural profiles.

A behavioural profile defines an abstraction of trace semantics of a net system. It is computed efficiently once certain structural and behavioural assumptions are satisfied. We presented formal results that allow for the computation of behavioural profiles for sound free-choice WF-systems in low polynomial time to the size of the system. This approach is completed by algorithms that compute behavioural profiles for the more generic class of bounded net systems at the expense of increased computational complexity. For all approaches, we presented experimental evaluations using three large model collections from industry.

Consistency Notions and Measures based on Behavioural Profiles

Using behavioural profiles, we introduced a spectrum of consistency notions to judge behaviour consistency of an alignment

between two net systems. In the line of these consistency notions, we also presented consistency measures that quantify behaviour consistency. This spectrum of notions and measures allows for gradually adapting the consistency requirements towards a dedicated setting. Further, we adjusted the consistency measures also for the use case of conformance checking of process logs.

We evaluated the consistency notions and measures from different angles. On the one hand, we were able to show empirically that certain consistency criteria based on behavioural profiles show a good approximation of the human consistency perception for a dedicated consistency setting. On the other hand, experimental results obtained for behaviour consistency between models of an industry reference model suggest that our measures are also suited in this context.

Consistency Management based on Behavioural Profiles.

Appropriate consistency notions and measures are only one facet of an overarching consistency management of related process models. With our work, we addressed various further dimensions. We presented an approach to propagate changes between aligned net systems. It aims at respecting the consistency requirements imposed by the presented notions and measures. Exploration of behavioural commonalities and differences has been addressed by the definition of a set algebra for behavioural profiles. It allows for computing with the abstracted behaviour of net systems. We reported on findings obtained with the set-algebraic operations for the alignments between models of an industry reference model. Further, we introduced an approach to model synthesis for behavioural profiles. Given a profile that meets certain assumptions, we presented an algorithm that yields an according net system. Finally, we proposed concepts to support root-cause analysis for non-conformance of a process log. The usefulness of these concepts has been shown in a case study with an international service provider.

We conclude that the contribution of this thesis is twofold. First, we introduced a novel behavioural model, a behavioural profile, which is more abstract than those commonly used for behavioural analysis. Besides the pure definition, we came up with supporting techniques ranging from the computation of behavioural profiles, over means to compute with them, to an approach to model synthesis. Second, we showed how this behavioural model is used to judge behaviour consistency of process models that capture the same business process. Many questions of behaviour consistency have been addressed before. To the best of

our knowledge, however, no work approached these questions following a relational approach as proposed in this thesis.

10.2 BEHAVIOURAL PROFILES IN THE BROADER CONTEXT

The definition of behavioural profiles, along with techniques that centre on this behavioural abstraction, is a contribution that is rather independent of the analysis of behaviour consistency. Actually, the notion of a behavioural profile already proved to be useful in a much broader context. Behavioural profiles provide a way to consider execution semantics in a lightweight manner. The relational semantics induced by behavioural profiles can be seen as a compromise between the accuracy of the behavioural model and the efficiency needed to reason with it. Behavioural profiles are computed efficiently for a broad class of net systems. Further, virtually all presented techniques that built on behavioural profiles do not require complex computations. Hence, behavioural profiles are well-suited whenever the implied abstraction is considered to be acceptable. Whether this is the case, clearly depends on the use case. In some scenarios, the implied abstraction is not harmful at all. For instance, if a technique is optimised by using behavioural profiles, the abstraction only limits the optimisation potential. In other scenarios, the implied abstraction may not be negligible. Nevertheless, the application of behavioural profiles allows for drawing initial conclusions efficiently. More accurate behavioural models that result in computationally hard analysis are then applied only on demand for a subset of the investigated models. As discussed in Section 9.7, the combination of conformance analysis based on behavioural profiles and based on a replay of cases would be an example for such a scenario.

In the remainder of this section, we briefly review applications of behavioural profiles. These applications illustrate a wide variety of use cases in the area of Business Process Management (BPM). However, we do not consider the application of behavioural profiles to be limited to the field of BPM.

Efficient Process Model Search

In Section 7.2, we introduced consistency measures based on behavioural profiles for an alignment between related process models. When reviewing the properties of these measures, we highlighted that they are not suited to judge behavioural similarity of process models. However, in recent work, we took up the ideas of these consistency measures to define behavioural similarity metrics based on behavioural profiles [254, 253]. In essence, these metrics quantify process model similarity using

the Jaccard coefficient defined over the profile relations of two process models. Further, the hierarchy of behavioural relations as discussed in Section 8.2 is taken into account. Since these similarity measures are proper metrics, we have been able to combine them with efficient indexing techniques for process model search [253].

Modelling Support based on Glossaries

There are various aspects that impact the understandability of a process model. In particular, the labelling has a strong influence on the understandability [317]. Appropriate reuse and alignment of model labels, therefore, is a quality criterion of a single model and a model collection. One approach to achieve these goals is the application of a glossary, which provides a centralised terminology for a dedicated domain, in the course of modelling [391]. In [349, 350], we showed how such a glossary may be generated from an existing model collection, e. g., a reference model, and how it is leveraged for modelling support. Besides the terms, this glossary also considers behavioural dependencies between them. To capture these dependencies, we utilised behavioural profiles. The dependencies are used to assess labelling quality of existing models, and provide rich modelling support for the creation of new models. Consider, for instance, two labels that are exclusive to each other in many models of a model collection and, thus, in the glossary derived from this collection. Those labels should not appear together in a model that is newly created.

Action Patterns of Process Models

Another approach to extract knowledge hidden in large process model collections builds on the notion of action patterns [421, 424, 426]. Action patterns capture reusable chunks of actions. In essence, actions are the verbs of activity labels. However, one may also consider different levels of generalisation of these verbs [426] or varying sensitivity to business objects [424]. As such, action patterns capture process knowledge that is more generic than a domain-specific reference model, but more concrete than generic process patterns, such as the workflow patterns[1] [456]. Using the notion of an action, co-occurrence action patterns and behavioural action patterns are derived from a model collection by means of association rules mining. Co-occurrence action patterns capture actions that are often observed together. They allow conclusions to be drawn on *what* should be

1 See http://www.workflowpatterns.com for further details.

covered by a process model. Behavioural action patterns capture behavioural dependencies that define *how* the actions should be applied. Since these dependencies have to be indirect, actions are not necessarily direct successors, they have been captured using behavioural profiles. Also, mining of action patterns benefits from the methods for the efficient computation of behavioural profiles.

Detection of Configuration Issues and Run Time Violations

Behavioural profiles have also been used to support the automation of business processes. Before an executable process model is enacted using a workflow engine, an operational process model is configured, i. e., it is enriched with all details needed for execution. As part of that, a data binding is introduced that assigns create/read/update/delete (CRUD) operations on data objects to activities. The data flow defined by these operations may be contradicting with the control flow defined between activities, e. g., access to data that may have not been created before yields a missing data anomaly. To detect such inconsistencies the work reported in [387] leverages behavioural profiles. Even though not all inconsistencies may be detected due to the assumed behavioural abstraction, this approach provides efficient support for the configuration of process models.

Behavioural profiles also proved valuable to detect control-flow violations once a business process is executed in a complex event processing environment. If a process models is not enacted directly using workflow technology, supporting information systems may provide the flexibility to deviate from the normative model. In [509], we proposed an approach to derive monitoring queries that, applied to event streams signalling the process execution, identify control flow violations. In essence, these queries encode violations of the behavioural dependencies as imposed by the behavioural profile of a process model.

10.3 LIMITATIONS & FUTURE RESEARCH

Our framework to assess behaviour consistency spans various aspects, from the creation of an alignment over consistency notions and measures to techniques for consistency management. Still, we also have to reflect on some limitations of our approach. These limitations relate to the purpose of consistency analysis, consistency preserving operators, and further process modelling dimensions. In the following, we discuss these limitations and outline directions for future research.

The Purpose of Consistency Analysis

We argued that the variety of drivers of process modelling results in differences in the level of abstraction and the assumed perspective of process models, even if they capture the same business process, cf., Section 1.2. Each model has to be appropriate towards its purpose, which may result in mismatches. Despite this fact, the models shall respect a certain notion of behaviour consistency, a certain freedom of contradictions. We outlined that the need for consistency is inherent to establish a shared understanding among process stakeholders. Further, we illustrated the need to consider consistency by exemplary drivers of consistency analysis, i. e., validation, inter-model analysis, and change propagation, see Section 1.3.

We highlighted that the variety of drivers of process modelling and consistency analysis requires a relativistic assessment of behaviour consistency. To accommodate for this requirement, our framework comprises different notions and measures that define a spectrum of consistency criteria. However, we largely neglected the relation between these criteria and the purpose of consistency analysis. The only exception to this has been our investigation on the consistency perception of process analysts. This part clearly relates to two dedicated drivers of process modelling and consistency analysis for the purpose of validation. Apart from that, our framework provides the flexibility to adapt the consistency criteria towards a concrete setting. Still, we do not provide any guidance on how to do this adaptation given a dedicated purpose of consistency analysis. Most likely, only a subset of the presented criteria is appropriate for a certain kind of analysis. Hence, future work is required to investigate the relation between the drivers of consistency analysis and consistency requirements imposed by them. In the end, such research may yield a method that guides the usage of the presented techniques in a specific context.

Consistency Preserving Operators

The focus of our work has been on assessing behaviour consistency a posteriori, once process models have been created independent of each other. As we discussed in Section 6.1 and Section 6.4, this is a major conceptual difference compared to work on process views or model refinements. Further, we motivated the weak criteria of our consistency spectrum with the presence of non-hierarchical relations between process models serving different purposes. This raises the question which transformation operations – that go beyond well-known hierarchical refinements – actually preserve the presented consistency criteria.

Once such transformations are available, they can guide the evolution of process models. If an alignment between related process models was shown to be consistent, consistency preserving operators define how to best adapt the models without lowering the alignment consistency. Finally, such operators may also guide the creation of new process models, following a *consistent by design* paradigm. As a first step towards the definition of consistency preserving operators, transformations that preserve behaviour inheritance [33] may be adapted for the presented consistency criteria.

Towards Holistic Consistency Assessment

The presented framework is limited to the control flow perspective. Since an explicit coordination of activity execution is the very core of a process model, we argued that it is reasonable to first approach consistency analysis from this perspective. Albeit reasonable, neglecting data and resource assignments is a limitation of our work. Clearly, a holistic approach to consistency assessment has to take the data and resource perspective into account.

We foresee two directions to realise such an extension. First, the influence of data and resources on the control flow may be considered by including the respective modelling perspectives directly in the formal model. Apparently, the data and resource dependencies may affect the behavioural profile, e. g., activities that are not constraint by any control flow dependency may compete for a single resource. To this end, one can rely on work on net-based formalisations of object life-cycles [451] and data access semantics [26] or lift the presented techniques to high-level Petri nets, such as coloured Petri nets (CPNs) [220], workflow nets with data (WFD-nets) [419], or dual workflow nets (DFNs) [157]. Then, the presented analysis techniques can be applied directly.

Second, different types of data access and resource assignments of activities may be captured using a relational model, similar to the profile relations. One can imagine multiple mutually exclusive data access relations, e. g., representing the CRUD operations and the absence of any access, that partition the Cartesian product of activities and data objects. In the same vein, a partitioning of the Cartesian product of activities and resources may be defined by different types of assignment relations. Using such a relational model, we foresee that many of the techniques presented for the relations of behavioural profiles may be lifted to the relational model for data access and resource assignments.

[1] Unified Modeling Language: Superstructure. Version 2.1.2. Technical report, Object Management Group (OMG), November 2007. (Cited on page 22.)

[2] Business Process Model and Notation (BPMN) Version 2.0. Technical report, Object Management Group (OMG), January 2011. (Cited on pages 18 and 165.)

[3] Norris Syed Abdullah, Shazia W. Sadiq, and Marta Indulska. Emerging challenges in information systems research for regulatory compliance management. In Pernici [348], pages 251–265. ISBN 978-3-642-13093-9. (Cited on page 8.)

[4] Witold Abramowicz and Heinrich C. Mayr, editors. *Business Information Systems, 9th International Conference on Business Information Systems, BIS 2006, May 31 - June 2, 2006, Klagenfurt, Austria*, volume 85 of *LNI*, 2006. GI. ISBN 3-88579-179-X. (Cited on pages 306 and 317.)

[5] Rafael Accorsi and Claus Wonnemann. Auditing workflow executions against dataflow policies. In Witold Abramowicz and Robert Tolksdorf, editors, *BIS*, volume 47 of *Lecture Notes in Business Information Processing*, pages 207–217. Springer, 2010. ISBN 978-3-642-12813-4. (Cited on page 276.)

[6] Arya Adriansyah, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Towards robust conformance checking. In zur Muehlen and Su [539], pages 122–133. ISBN 978-3-642-20510-1. (Cited on pages 254 and 275.)

[7] Arya Adriansyah, Natalia Sidorova, and Boudewijn F. van Dongen. Cost-based fitness in conformance checking. In *Proceedings of the 11th International Conference on Application of Concurrency to System Design (ACSD 2011)*, 2011. To appear. (Cited on page 275.)

[8] Pär J. Ågerfalk and Owen Eriksson. Action-oriented conceptual modelling. *EJIS*, 13(1):80–92, 2004. (Cited on page 48.)

[9] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB*, pages 487–499. Morgan Kaufmann, 1994. ISBN 1-55860-153-8. (Cited on page 268.)

[10] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *SIGMOD Conference*, pages 207–216. ACM Press, 1993. (Cited on page 268.)

[11] Rakesh Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workflow logs. In Hans-Jörg Schek, Fèlix Saltor, Isidro Ramos, and Gustavo Alonso, editors, *EDBT*, volume 1377 of *Lecture Notes in Computer Science*, pages 469–483. Springer, 1998. ISBN 3-540-64264-1. (Cited on pages 88, 248, and 275.)

[12] Gail-Joon Ahn and Ravi S. Sandhu. Role-based authorization constraints specification. *ACM Trans. Inf. Syst. Secur.*, 3(4):207–226, 2000. (Cited on page 276.)

[13] Alfred V. Aho and Jeffrey D. Ullman. *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1972. ISBN 0-13-914556-7. (Cited on page 99.)

[14] Alfred V. Aho and Jeffrey D. Ullman. *Foundations of Computer Science*. Computer Science Press, New York, W. Freeman and Company, 1995. (Cited on page 4.)

[15] Marco Aiello, Mikio Aoyama, Francisco Curbera, and Mike P. Papazoglou, editors. *Service-Oriented Computing - ICSOC 2004, Second International Conference, New York, NY, USA, November 15-19, 2004, Proceedings*, 2004. ACM. ISBN 1-58113-871-7. (Cited on pages 298 and 302.)

[16] Manoli Albert, Javier Muñoz, Vicente Pelechano, and Oscar Pastor. Model to text transformation in practice: Generating code from rich associations specifications. In John F. Roddick, V. Richard Benjamins, Samira Si-Said Cherfi, Roger H. L. Chiang, Christophe Claramunt, Ramez Elmasri, Fabio Grandi, Hyoil Han, Martin Hepp, Miltiadis D. Lytras, Vojislav B. Misic, Geert Poels, Il-Yeol Song, Juan Trujillo, and Christelle Vangenot, editors, *ER (Workshops)*, volume 4231 of *Lecture Notes in Computer Science*, pages 63–72. Springer, 2006. ISBN 3-540-47703-9. (Cited on page 60.)

[17] Alexandre Alves et al. Web Services Business Process Execution Language Version 2.0. Technical report, OASIS, January 2007. URL `www.oasis-open.org/committees/wsbpel/`. (Cited on page 24.)

[18] Thomas Allweyer. *BPMN 2.0: Introduction to the Standard for Business Process Modeling*. Books on Demand, 2010. (Cited on page 19.)

[19] Gustavo Alonso, Peter Dadam, and Michael Rosemann, editors. *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*, volume 4714 of *Lecture Notes in Computer Science*, 2007. Springer. ISBN 978-3-540-75182-3. (Cited on pages 290, 291, 301, 306, 308, 310, 316, and 325.)

[20] Stephen Alstrup, Dov Harel, Peter W. Lauridsen, and Mikkel Thorup. Dominators in linear time. *SIAM J. Comput.*, 28(6):2117–2132, 1999. (Cited on page 103.)

[21] Birger Andersson, Maria Bergholtz, Ananda Edirisuriya, Tharaka Ilayperuma, and Paul Johannesson. A declarative foundation of process models. In Pastor and e Cunha [345], pages 233–247. ISBN 3-540-26095-1. (Cited on pages 49, 155, 172, and 176.)

[22] Birger Andersson, Maria Bergholtz, Bertrand Grégoire, Paul Johannesson, Michael Schmitt, and Jelena Zdravkovic. From business to process models - a chaining methodology. In Pigneur and Woo [355]. (Cited on page 49.)

[23] Birger Andersson, Paul Johannesson, and Jelena Zdravkovic. Aligning goals and services through goal and business modelling. *Inf. Syst. E-Business Management*, 7(2):143–169, 2009. (Cited on pages 42 and 49.)

[24] Danilo Ardagna, Massimo Mecella, and Jian Yang, editors. *Business Process Management Workshops, BPM 2008 International Workshops, Milano, Italy, September 1-4, 2008. Revised Papers*, volume 17 of *Lecture Notes in Business Information Processing*, 2009. Springer. ISBN 978-3-642-00327-1. (Cited on pages 293 and 301.)

[25] Ahmed Awad, Gero Decker, and Mathias Weske. Efficient compliance checking using BPMN-Q and temporal logic. In Dumas et al. [135], pages 326–341. ISBN 978-3-540-85757-0. (Cited on page 42.)

[26] Ahmed Awad, Gero Decker, and Niels Lohmann. Diagnosing and repairing data anomalies in process models. In Rinderle-Ma et al. [385], pages 5–16. ISBN 978-3-642-12185-2. (Cited on pages 35 and 286.)

[27] Ahmed Awad, Matthias Weidlich, and Mathias Weske. Visually specifying compliance rules and explaining their violations for business processes. *J. Vis. Lang. Comput.*, 22(1):30–55, 2011. (Cited on pages 8 and 42.)

[28] Felix Bachmann and Leonard J. Bass. Managing variability in software architectures. In *SSR*, pages 126–132, 2001. (Cited on page 245.)

[29] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, Addison-Wesley, New York, 1999. (Cited on page 65.)

[30] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. ISBN 978-0-262-02649-9. (Cited on page 152.)

[31] Wasana Bandara, Guy G. Gable, and Michael Rosemann. Factors and measures of business process modelling: model building through a multiple case study. *EJIS*, 14(4):347–360, 2005. (Cited on pages 9 and 47.)

[32] Victor R. Basili, Lionel C. Briand, and Walcélio L. Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Trans. Software Eng.*, 22(10):751–761, 1996. (Cited on page 163.)

[33] Twan Basten and Wil M. P. van der Aalst. Inheritance of behavior. *J. Log. Algebr. Program.*, 47(2):47–145, 2001. (Cited on pages 152, 172, 191, 192, 218, 244, and 286.)

[34] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364, 1986. (Cited on pages 6 and 47.)

[35] Don S. Batory, Clay Johnson, Bob MacDonald, and Dale von Heeder. Achieving extensibility through product-lines and domain-specific languages: A case study. *ACM Trans. Softw. Eng. Methodol.*, 11(2):191–214, 2002. (Cited on page 245.)

[36] Don S. Batory, Jacob Neal Sarvela, and Axel Rauschmayer. Scaling stepwise refinement. *IEEE Trans. Software Eng.*, 30(6):355–371, 2004. (Cited on page 246.)

[37] Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with spqr-trees. *Algorithmica*, 15(4):302–318, 1996. (Cited on pages 109 and 116.)

[38] Jörg Becker, Michael Rosemann, and Christoph von Uthmann. Guidelines of business process modeling. In van der Aalst et al. [453], pages 30–49. ISBN 3-540-67454-3. (Cited on page 50.)

[39] Jörg Becker, Patrick Delfmann, Sebastian Herwig, Lukasz Lis, and Armin Stein. Formalizing linguistic conventions for conceptual models. In Laender et al. [260], pages 70–83. ISBN 978-3-642-04839-5. (Cited on page 48.)

[40] Jörg Becker, Martin Kugeler, and Michael Rosemann, editors. *Process Management: A Guide for the Design of Business Processes*. Springer, Berlin, 2003. (Cited on pages 5, 6, and 17.)

[41] Jörg Becker, Patrick Delfmann, and Ralf Knackstedt. Adaptive reference modeling: Integrating configurative and generic adaptation techniques for information models. In Jörg Becker and Patrick Delfmann, editors, *Reference Modeling*, pages 27–58. Physica-Verlag HD, 2007. ISBN 978-3-7908-1966-3. URL http://dx.doi.org/10.1007/978-3-7908-1966-3_2. (Cited on page 246.)

[42] Zohra Bellahsene and Michel Léonard, editors. *Advanced Information Systems Engineering, 20th International Conference, CAiSE 2008, Montpellier, France, June 16-20, 2008, Proceedings*, volume 5074 of *Lecture Notes in Computer Science*, 2008. Springer. ISBN 978-3-540-69533-2. (Cited on pages 322 and 327.)

[43] Boualem Benatallah, Fabio Casati, and Farouk Toumani. Representing, analysing and managing web service protocols. *Data Knowl. Eng.*, 58(3): 327–357, 2006. (Cited on page 177.)

[44] Massimo Benerecetti, Paolo Bouquet, and Chiara Ghidini. On the dimensions of context dependence: Partiality, approximation, and perspective. In Varol Akman, Paolo Bouquet, Richmond H. Thomason, and Roger A. Young, editors, *CONTEXT*, volume 2116 of *Lecture Notes in Computer Science*, pages 59–72. Springer, 2001. ISBN 3-540-42379-6. (Cited on page 48.)

[45] Robin Bergenthum, Jörg Desel, Robert Lorenz, and Sebastian Mauser. Process mining based on regions of languages. In Alonso et al. [19], pages 375–383. ISBN 978-3-540-75182-3. (Cited on page 249.)

[46] Maria Bergholtz and Paul Johannesson. Classifying the semantics of relationships in conceptual modeling by categorization of roles. In Ana María Moreno and Reind P. van de Riet, editors, *NLDB*, volume 3 of *LNI*, pages 199–203. GI, 2001. ISBN 3-88579-332-6. (Cited on page 48.)

[47] Abraham Bernstein and Mark Klein. Towards high-precision service retrieval. In Horrocks and Hendler [215], pages 84–101. ISBN 3-540-43760-6. (Cited on page 51.)

[48] Elisa Bertino, Elena Ferrari, and Vijayalakshmi Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.*, 2(1):65–104, 1999. (Cited on page 276.)

[49] Eike Best. Structure theory of Petri nets: The free choice hiatus. In Brauer et al. [63], pages 168–205. ISBN 3-540-17905-4. (Cited on page 33.)

[50] Eike Best and Raymond R. Devillers. Sequential and concurrent behaviour in Petri net theory. *Theor. Comput. Sci.*, 55(1):87–136, 1987. (Cited on page 86.)

[51] Eike Best, Raymond R. Devillers, Astrid Kiehn, and Lucia Pomello. Concurrent bisimulations in Petri nets. *Acta Inf.*, 28(3):231–264, 1991. (Cited on page 230.)

[52] Jean Bézivin. On the unification power of models. *Software and System Modeling*, 4(2):171–188, 2005. (Cited on page 4.)

[53] Jean Bézivin and Olivier Gerbé. Towards a precise definition of the OMG/MDA framework. In *ASE*, pages 273–280. IEEE Computer Society, 2001. ISBN 0-7695-1426-X. (Cited on page 4.)

[54] Kamal Bhattacharya, Richard Hull, and Jianwen Su. *Handbook of Research on Business Process Modeling*, chapter A data-centric design methodology for business processes, pages 503–531. Idea Group Reference, 2009. (Cited on pages 11 and 18.)

[55] Ralph Bobrik, Manfred Reichert, and Thomas Bauer. View-based process visualization. In Alonso et al. [19], pages 88–95. ISBN 978-3-540-75182-3. (Cited on page 176.)

[56] Eerke A. Boiten, Howard Bowman, John Derrick, Peter F. Linington, and Maarten Steen. Viewpoint consistency in odp. *Computer Networks*, 34(3): 503–537, 2000. (Cited on page 176.)

[57] Egon Börger and Bernhard Thalheim. Modeling workflows, interaction patterns, web services and business processes: The ASM-based approach. In Egon Börger, Michael J. Butler, Jonathan P. Bowen, and Paul Boca, editors, *ABZ*, volume 5238 of *Lecture Notes in Computer Science*, pages 24–38. Springer, 2008. ISBN 978-3-540-87602-1. (Cited on page 19.)

[58] Egon Börger, Alessandra Cavarra, and Elvinia Riccobene. An ASM semantics for UML activity diagrams. In Teodor Rus, editor, *AMAST*, volume 1816 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2000. ISBN 3-540-67530-2. (Cited on page 23.)

[59] R. P. Jagadeesh Chandra Bose and Wil M. P. van der Aalst. Trace alignment in process mining: Opportunities for process diagnostics. In Hull et al. [217], pages 227–242. ISBN 978-3-642-15617-5. (Cited on page 276.)

[60] Athman Bouguettaya, Ingolf Krüger, and Tiziana Margaria, editors. *Service-Oriented Computing - ICSOC 2008, 6th International Conference, Sydney, Australia, December 1-5, 2008. Proceedings*, volume 5364 of *Lecture Notes in Computer Science*, 2008. ISBN 978-3-540-89647-0. (Cited on pages 322 and 326.)

[61] Howard Bowman, Maarten Steen, Eerke A. Boiten, and John Derrick. A formal framework for viewpoint consistency. *Formal Methods in System Design*, 21(2):111–166, 2002. (Cited on page 176.)

[62] George E. P. Box. Robustness in the strategy of scientific model building. In *Robustness in Statistics: Proceedings of a Workshop*, New York, 1979. Academic Press. (Cited on page 3.)

[63] Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors. *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986*, volume 254 of *Lecture Notes in Computer Science*, 1987. Springer. ISBN 3-540-17905-4. (Cited on pages 290 and 303.)

[64] Wilfried Brauer, Robert Gold, and Walter Vogler. A survey of behaviour and equivalence preserving refinements of Petri nets. In Rozenberg [394], pages 1–46. ISBN 3-540-53863-1. (Cited on page 152.)

[65] Mario Bravetti and Tevfik Bultan, editors. *Web Services and Formal Methods - 7th International Workshop, WS-FM 2010, Hoboken, NJ, USA, September 16-17, 2010. Revised Selected Papers*, volume 6551 of *Lecture Notes in Computer Science*, 2011. Springer. ISBN 978-3-642-19588-4. (Cited on pages 313 and 324.)

[66] Mario Bravetti, Manuel Núñez, and Gianluigi Zavattaro, editors. *Web Services and Formal Methods, Third International Workshop, WS-FM 2006 Vienna, Austria, September 8-9, 2006, Proceedings*, volume 4184 of *Lecture Notes in Computer Science*, 2006. Springer. ISBN 3-540-38862-1. (Cited on pages 295, 320, and 326.)

[67] Eric Brill. A simple rule-based part of speech tagger. In *ANLP*, pages 152–155, 1992. (Cited on page 50.)

[68] Saartje Brockmans, Marc Ehrig, Agnes Koschmider, Andreas Oberweis, and Rudi Studer. Semantic alignment of business processes. In Yannis Manolopoulos, Joaquim Filipe, Panos Constantopoulos, and José Cordeiro, editors, *ICEIS (3)*, pages 191–196, 2006. ISBN 972-8865-41-4. (Cited on page 51.)

[69] Michael C. Browne, Edmund M. Clarke, and Orna Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theor. Comput. Sci.*, 59:115–131, 1988. (Cited on pages 152 and 153.)

[70] Janis A. Bubenko Jr. and Benkt Wangler. Objective driven capture of business rules and of information systems requirements. In *Proceedings of the IEEE Systems Man and Cybernetics '93 Conference*, pages 670–677, Le Touquet, France, October 1993. (Cited on pages 3, 7, 49, and 163.)

[71] Adam L. Buchsbaum, Haim Kaplan, Anne Rogers, and Jeffery Westbrook. A new, simpler linear-time dominators algorithm. *ACM Trans. Program. Lang. Syst.*, 20(6):1265–1296, 1998. (Cited on page 103.)

[72] Alexander Budanitsky and Graeme Hirst. Evaluating WordNet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32 (1):13–47, 2006. (Cited on page 50.)

[73] Horst Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997. (Cited on page 51.)

[74] Horst Bunke. Graph matching: Theoretical foundations, algorithms, and applications. In *Proceedings of the International Conference on Vision Interface*, pages 82–88, Montreal, Quebec, Canada, May 2000. (Cited on pages 51 and 64.)

[75] Horst Bunke and Gudrun Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245 – 253, 1983. ISSN 0167-8655. URL http://www.sciencedirect.com/science/article/B6V15-48MPV00-1K/2/6f816d072c71e50b1a80858a8b488463. (Cited on pages 51 and 64.)

[76] J. F. M. Burg and Reind P. van de Riet. The impact of linguistics on conceptual models: Consistency and understandability. *Data Knowl. Eng.*, 21(2):131–146, 1997. (Cited on page 48.)

[77] Andrew Burton-Jones, Yair Wand, and Ron Weber. Guidelines for empirical evaluations of conceptual modeling grammars. *J. AIS*, 10(6), 2009. (Cited on page 164.)

[78] Christoph Bussler. *B2B Integration: Concepts and Architecture*. Springer, 2003. (Cited on page 8.)

[79] Carlos Canal, Ernesto Pimentel, and José M. Troya. Compatibility and inheritance in software architectures. *Sci. Comput. Program.*, 41(2):105–138, 2001. (Cited on page 177.)

[80] S. N. Cant, D. R. Jeffery, and B. Henderson-Sellers. A conceptual model of cognitive complexity of elements of the programming process. *Information and Software Technology*, 37(7):351–362, 1995. (Cited on page 209.)

[81] Jorge Cardoso. Business process control-flow complexity: Metric, evaluation, and validation. *Int. J. Web Service Res.*, 5(2):49–76, 2008. (Cited on page 210.)

[82] Jorge Cardoso, Jan Mendling, Gustaf Neumann, and Hajo A. Reijers. A discourse on complexity of process models. In Eder and Dustdar [139], pages 117–128. ISBN 3-540-38444-8. (Cited on page 209.)

[83] Josep Carmona, Jordi Cortadella, and Michael Kishinevsky. New region-based algorithms for deriving bounded Petri nets. *IEEE Trans. Computers*, 59(3):371–384, 2010. (Cited on page 249.)

[84] Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors. *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, 2006. ACM. ISBN 1-59593-323-9. (Cited on pages 303 and 314.)

[85] Namyoun Choi, Il-Yeol Song, and Hyoil Han. A survey on ontology mapping. *SIGMOD Record*, 35(3):34–41, 2006. (Cited on pages 42 and 47.)

[86] María Agustina Cibrán. Translating BPMN models into UML activities. In Ardagna et al. [24], pages 236–247. ISBN 978-3-642-00327-1. (Cited on pages 24 and 47.)

[87] Marcus Ciolkowski, Oliver Laitenberger, Sira Vegas, and Stefan Biffl. Practical experiences in the design and conduct of surveys in empirical software engineering. In Reidar Conradi and Alf Inge Wang, editors, *ESERNET*, volume 2765 of *Lecture Notes in Computer Science*, pages 104–128. Springer, 2003. ISBN 3-540-40672-7. (Cited on page 165.)

[88] William W. Cohen, Pradeep D. Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In Subbarao Kambhampati and Craig A. Knoblock, editors, *IIWeb*, pages 73–78, 2003. (Cited on page 50.)

[89] David Cohn and Richard Hull. Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Data Eng. Bull.*, 32(3):3–9, 2009. (Cited on pages 11 and 18.)

[90] Carlo Combi and Roberto Posenato. Controllability in temporal conceptual workflow schemata. In Dayal et al. [103], pages 64–79. ISBN 978-3-642-03847-1. (Cited on page 230.)

[91] Juan Carlos Corrales, Daniela Grigori, and Mokrane Bouzeghoub. BPEL processes matchmaking for service discovery. In Meersman and Tari [306], pages 237–254. ISBN 3-540-48287-3. (Cited on pages 50, 51, 52, 53, and 54.)

[92] Jordi Cortadella, Michael Kishinevsky, Luciano Lavagno, and Alexandre Yakovlev. Deriving Petri nets for finite transition systems. *IEEE Trans. Computers*, 47(8):859–882, 1998. (Cited on page 249.)

[93] Jordi Cortadella, Alexandre Yakovlev, and Grzegorz Rozenberg, editors. *Concurrency and Hardware Design, Advances in Petri Nets*, volume 2549 of *Lecture Notes in Computer Science*, 2002. Springer. ISBN 3-540-00199-9. (Cited on page 28.)

[94] Jason Crampton and Hemanth Khambhammettu. Delegation and satisfiability in workflow systems. In Indrakshi Ray and Ninghui Li, editors, *SACMAT*, pages 31–40. ACM, 2008. ISBN 978-1-60558-129-3. (Cited on page 276.)

[95] Thomas A. Curran, Gerhard Keller, and Andrew Ladd. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Prentice Hall, 1997. (Cited on pages 135, 190, and 241.)

[96] Krzysztof Czarnecki and Michal Antkiewicz. Mapping features to models: A template approach based on superimposed variants. In Robert Glück and Michael R. Lowry, editors, *GPCE*, volume 3676 of *Lecture Notes in Computer Science*, pages 422–437. Springer, 2005. ISBN 3-540-29138-5. (Cited on page 246.)

[97] Peter Dadam and Manfred Reichert. The ADEPT project: a decade of research and development for robust and flexible process support. *Computer Science - R&D*, 23(2):81–97, 2009. (Cited on pages 144, 213, and 245.)

[98] Maya Daneva, Ralf Heib, and August-Wilhelm Scheer. Benchmarking business process models. Technical report, IWi Report 136, Saarland University, 1996. (Cited on page 210.)

[99] Anindya Datta. Automating the discovery of as-is business process models: Probabilistic and algorithmic approaches. *Information Systems Research*, 9(3):275–301, 1998. (Cited on pages 88, 248, and 275.)

[100] Thomas H. Davenport. *Process Innovation: Reengineering Work Through Information Technology*. Harvard Business, 1992. (Cited on pages 3, 5, 7, and 49.)

[101] Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2nd edition, 2002. (Cited on page 75.)

[102] Islay Davies, Peter F. Green, Michael Rosemann, Marta Indulska, and Stan Gallo. How do practitioners use conceptual modeling in practice? *Data Knowl. Eng.*, 58(3):358–380, 2006. (Cited on pages 11 and 18.)

[103] Umeshwar Dayal, Johann Eder, Jana Koehler, and Hajo A. Reijers, editors. *Business Process Management, 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings*, volume 5701 of *Lecture Notes in Computer Science*, 2009. Springer. ISBN 978-3-642-03847-1. (Cited on pages 293, 296, 298, 303, 307, and 313.)

[104] Ana Karla A. de Medeiros, Wil M. P. van der Aalst, and A. J. M. M. Weijters. Workflow mining: Current status and future directions. In Meersman et al. [308], pages 389–406. ISBN 3-540-20498-9. (Cited on pages 90 and 248.)

[105] Ana Karla A. de Medeiros, Boudewijn F. van Dongen, Wil M. P. van der Aalst, and A. J. M. M. Weijters. Process mining for ubiquitous mobile systems: An overview and a concrete algorithm. In Luciano Baresi, Schahram Dustdar, Harald Gall, and Maristella Matera, editors, *UMICS*, volume 3272 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 2004. ISBN 3-540-24100-0. (Cited on pages 90 and 248.)

[106] Ana Karla A. de Medeiros, A. J. M. M. Weijters, and Wil M. P. van der Aalst. Genetic process mining: an experimental evaluation. *Data Min. Knowl. Discov.*, 14(2):245–304, 2007. (Cited on pages 90, 144, 211, 248, and 250.)

[107] Ana Karla Alves de Medeiros, Wil M. P. van der Aalst, and A. J. M. M. Weijters. Quantifying process equivalence based on observed behavior. *Data Knowl. Eng.*, 64(1):55–74, 2008. (Cited on pages 181, 211, 254, 272, and 275.)

[108] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of A*. *J. ACM*, 32(3):505–536, 1985. (Cited on pages 63 and 203.)

[109] Gero Decker and Jan Mendling. Process instantiation. *Data Knowl. Eng.*, 68(9):777–792, 2009. (Cited on pages 22, 37, 136, and 143.)

[110] Gero Decker and Mathias Weske. Behavioral consistency for B2B process integration. In Krogstie et al. [249], pages 81–95. ISBN 978-3-540-72987-7. (Cited on page 177.)

[111] Gero Decker and Mathias Weske. Instance isolation analysis for service-oriented architectures. In *IEEE SCC (1)*, pages 249–256. IEEE Computer Society, 2008. ISBN 978-0-7695-3283-7. (Cited on page 39.)

[112] Gero Decker and Mathias Weske. Interaction-centric modeling of process choreographies. *Inf. Syst.*, 36(2):292–312, 2011. (Cited on pages 88 and 177.)

[113] Gero Decker, Johannes Maria Zaha, and Marlon Dumas. Execution semantics for service choreographies. In Bravetti et al. [66], pages 163–177. ISBN 3-540-38862-1. (Cited on page 88.)

[114] Gero Decker, Willi Tscheschner, and Jörg Puchan. Migration von EPK zu BPMN. In Markus Nüttgens, Frank J. Rump, Jan Mendling, and Nick Gehrke, editors, *GI-Workshop Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (EPK)*, volume 554 of *WS-CEUR*, pages 91–109, Berlin, Germany, November 2009. In German. (Cited on pages 22 and 47.)

[115] Ken Decreus, Monique Snoeck, and Geert Poels. Practical challenges for methods transforming i* goal models into business process models. In *RE*, pages 15–23. IEEE Computer Society, 2009. ISBN 978-0-7695-3761-0. (Cited on page 49.)

[116] Juliane Dehnert and Wil M. P. van der Aalst. Bridging the gap between business models and workflow specifications. *Int. J. Cooperative Inf. Syst.*, 13(3):289–332, 2004. (Cited on pages 22, 35, 38, and 175.)

[117] Lois M. L. Delcambre, Christian Kop, Heinrich C. Mayr, John Mylopoulos, and Oscar Pastor, editors. *Conceptual Modeling - ER 2005, 24th International Conference on Conceptual Modeling, Klagenfurt, Austria, October 24-28, 2005, Proceedings*, volume 3716 of *Lecture Notes in Computer Science*, 2005. Springer. ISBN 3-540-29389-2. (Cited on pages 316 and 325.)

[118] Zsófia Derzsi and Jaap Gordijn. A framework for business/IT alignment in networked value constellations. In Pigneur and Woo [355]. (Cited on page 49.)

[119] Jörg Desel and Javier Esparza. *Free-Choice Petri Nets*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1995. (Cited on pages 28, 33, and 35.)

[120] Robin Dhamankar, Yoonkyong Lee, AnHai Doan, Alon Y. Halevy, and Pedro Domingos. iMAP: Discovering complex mappings between database schemas. In Gerhard Weikum, Arnd Christian König, and Stefan Deßloch, editors, *SIGMOD Conference*, pages 383–394. ACM, 2004. ISBN 1-58113-859-8. (Cited on pages 54 and 57.)

[121] Volker Diekert and Grzegorz Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995. (Cited on page 32.)

[122] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 1997. (Cited on page 58.)

[123] Remco M. Dijkman. A classification of differences between similar business processes. In *EDOC*, pages 37–50. IEEE Computer Society, 2007. (Cited on pages 49, 51, 54, 166, 174, and 212.)

[124] Remco M. Dijkman. Diagnosing differences between business process models. In Dumas et al. [135], pages 261–277. ISBN 978-3-540-85757-0. (Cited on page 212.)

[125] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Information & Software Technology*, 50(12):1281–1294, 2008. (Cited on pages 19, 35, and 37.)

[126] Remco M. Dijkman, Dick A. C. Quartel, and Marten van Sinderen. Consistency in multi-viewpoint design of enterprise information systems. *Information & Software Technology*, 50(7-8):737–752, 2008. (Cited on page 176.)

[127] Remco M. Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph matching algorithms for business process model similarity search. In Dayal et al. [103], pages 48–63. ISBN 978-3-642-03847-1. (Cited on pages 50, 52, 63, and 64.)

[128] Remco M. Dijkman, Marlon Dumas, Luciano García-Bañuelos, and Reina Käärik. Aligning business process models. In *EDOC*, pages 45–53. IEEE Computer Society, 2009. ISBN 978-0-7695-3785-6. (Cited on pages 52, 64, 66, and 67.)

[129] Remco M. Dijkman, Marlon Dumas, Boudewijn F. van Dongen, Reina Käärik, and Jan Mendling. Similarity of business process models: Metrics and evaluation. *Inf. Syst.*, 36(2):498–516, 2011. (Cited on pages 50, 51, and 52.)

[130] Hong Hai Do, Sergey Melnik, and Erhard Rahm. Comparison of schema matching evaluations. In Akmal B. Chaudhri, Mario Jeckle, Erhard Rahm, and Rainer Unland, editors, *Web, Web-Services, and Database Systems*, volume 2593 of *Lecture Notes in Computer Science*, pages 221–237. Springer, 2002. ISBN 3-540-00745-8. (Cited on page 66.)

[131] AnHai Doan and Alon Y. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine*, 26(1):83–94, 2005. (Cited on pages 42 and 47.)

[132] Yurdaer N. Doganata and Francisco Curbera. Effect of using automated auditing tools on detecting compliance failures in unmanaged processes. In Dayal et al. [103], pages 310–326. ISBN 978-3-642-03847-1. (Cited on page 42.)

[133] Eric Dubois and Klaus Pohl, editors. *Advanced Information Systems Engineering, 18th International Conference, CAiSE 2006, Luxembourg, Luxembourg, June 5-9, 2006, Proceedings*, volume 4001 of *Lecture Notes in Computer Science*, 2006. Springer. ISBN 3-540-34652-X. (Cited on pages 312 and 316.)

[134] Marlon Dumas, Wil M. van der Aalst, and Arthur H. ter Hofstede, editors. *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. John Wiley & Sons, 2005. (Cited on page 7.)

[135] Marlon Dumas, Manfred Reichert, and Ming-Chien Shan, editors. *Business Process Management, 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008. Proceedings*, volume 5240 of *Lecture Notes in Computer Science*, 2008. Springer. ISBN 978-3-540-85757-0. (Cited on pages 289, 296, 300, and 305.)

[136] Marlon Dumas, Luciano García-Bañuelos, and Remco M. Dijkman. Similarity search of business process models. *IEEE Data Eng. Bull.*, 32(3): 23–28, 2009. (Cited on pages 50, 53, 181, and 211.)

[137] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in property specifications for finite-state verification. In *ICSE*, pages 411–420, 1999. (Cited on page 79.)

[138] Jürgen Ebert and Gregor Engels. Observable or invocable behaviour - you have to choose! Technical report, University of Koblenz, Germany, 1994. (Cited on pages 152, 218, and 244.)

[139] Johann Eder and Schahram Dustdar, editors. *Business Process Management Workshops, BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, Vienna, Austria, September 4-7, 2006, Proceedings*, volume 4103 of *Lecture Notes in Computer Science*, 2006. Springer. ISBN 3-540-38444-8. (Cited on pages 293, 300, 304, and 310.)

[140] Henk Eertink, Wil Janssen, Paul Oude Luttighuis, Wouter B. Teeuw, and Chris A. Vissers. A business process design language. In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *World Congress on Formal Methods*, volume 1708 of *Lecture Notes in Computer Science*, pages 76–95. Springer, 1999. ISBN 3-540-66587-0. (Cited on page 88.)

[141] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Partial (set) 2-structures. Part I: Basic notions and the representation problem, Part II: State spaces of concurrent systems. *Acta Inf.*, 27(4):315–368, 1989. (Cited on page 249.)

[142] Marc Ehrig, Agnes Koschmider, and Andreas Oberweis. Measuring similarity between semantic business process models. In John F. Roddick and Annika Hinze, editors, *APCCM*, volume 67 of *CRPIT*, pages 71–80. Australian Computer Society, 2007. ISBN 1-920-68248-1. (Cited on pages 50 and 51.)

[143] Hans-Erik Eriksson and Magnus Penker. *Business Modelling with UML: Business Patterns at Work*. John Wiley & Sons, 2000. (Cited on page 23.)

[144] Rik Eshuis and Paul W. P. J. Grefen. Structural matching of BPEL processes. In *ECOWS*, pages 171–180. IEEE Computer Society, 2007. (Cited on pages 50, 53, 87, and 144.)

[145] Rik Eshuis and Paul W. P. J. Grefen. Constructing customized process views. *Data Knowl. Eng.*, 64(2):419–438, 2008. (Cited on page 176.)

[146] Rik Eshuis and Roel Wieringa. A real-time execution semantics for UML activity diagrams. In Heinrich Hußmann, editor, *FASE*, volume 2029 of *Lecture Notes in Computer Science*, pages 76–90. Springer, 2001. ISBN 3-540-41863-6. (Cited on page 23.)

[147] Javier Esparza. A polynomial-time algorithm for checking consistency of free-choice signal transition graphs. *Fundam. Inform.*, 62(2):197–220, 2004. (Cited on page 97.)

[148] Javier Esparza and Keijo Heljanko. A new unfolding approach to LTL model checking. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 475–486. Springer, 2000. ISBN 3-540-67715-1. (Cited on page 145.)

[149] Javier Esparza and Keijo Heljanko. *Unfoldings: A Partial-Order Approach to Model Checking*. Springer Berlin, 2008. (Cited on pages 118, 119, 120, 140, and 145.)

[150] Javier Esparza and Manuel Silva. Circuits, handles, bridges and nets. In Rozenberg [394], pages 210–242. ISBN 3-540-53863-1. (Cited on page 108.)

[151] Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design*, 20(3): 285–310, 2002. (Cited on pages 119, 120, 127, 134, and 140.)

[152] Sandro Etalle and William H. Winsborough. A posteriori compliance control. In Volkmar Lotz and Bhavani M. Thuraisingham, editors, *SACMAT*, pages 11–20. ACM, 2007. ISBN 978-1-59593-745-2. (Cited on page 276.)

[153] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer, Heidelberg (DE), 2007. ISBN 3-540-49611-4. (Cited on pages 42, 44, 46, 47, 48, 50, and 187.)

[154] Joerg Evermann. Theories of meaning in schema matching: An exploratory study. *Inf. Syst.*, 34(1):28–44, 2009. (Cited on page 163.)

[155] Dirk Fahland and Wolfgang Reisig. ASM-based semantics for BPEL: The negative control flow. In *Abstract State Machines*, pages 131–152, 2005. (Cited on page 25.)

[156] Dirk Fahland, Cédric Favre, Barbara Jobstmann, Jana Koehler, Niels Lohmann, Hagen Völzer, and Karsten Wolf. Instantaneous soundness checking of industrial business process models. In Dayal et al. [103], pages 278–293. ISBN 978-3-642-03847-1. (Cited on page 139.)

[157] Shaokun Fan, Wan-Chun Dou, and Jinjun Chen. Dual workflow nets: Mixed control/data-flow representation for workflow modeling and verification. In Kevin Chen-Chuan Chang, Wei Wang, Lei Chen 0002, Clarence A. Ellis, Ching-Hsien Hsu, Ah Chung Tsoi, and Haixun Wang, editors, *APWeb/WAIM Workshops*, volume 4537 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 2007. ISBN 978-3-540-72908-2. (Cited on pages 36 and 286.)

[158] Roozbeh Farahbod, Uwe Glässer, and Mona Vajihollahi. Specification and validation of the business process execution language for web services. In Wolf Zimmermann and Bernhard Thalheim, editors, *Abstract State Machines*, volume 3052 of *Lecture Notes in Computer Science*, pages 78–94. Springer, 2004. ISBN 3-540-22094-1. (Cited on page 25.)

[159] David F. Ferraiolo, John F. Barkley, and D. Richard Kuhn. A role-based access control model and reference implementation within a corporate intranet. *ACM Trans. Inf. Syst. Secur.*, 2(1):34–64, 1999. (Cited on page 276.)

[160] David F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001. (Cited on page 276.)

[161] Andrea Ferrara. Web services: a process algebra approach. In Aiello et al. [15], pages 242–251. ISBN 1-58113-871-7. (Cited on page 25.)

[162] Christian Fillies, Gay Wood-Albrecht, and Frauke Weichhardt. Pragmatic applications of the semantic web using SemTalk. *Computer Networks*, 42(5):599–615, 2003. (Cited on page 48.)

[163] Anthony Finkelstein, Dov M. Gabbay, Anthony Hunter, Jeff Kramer, and Bashar Nuseibeh. Inconsistency handling in multperspective specifications. *IEEE Trans. Software Eng.*, 20(8):569–578, 1994. (Cited on pages 176 and 177.)

[164] Jakob Freund and Bernd Rücker. *Praxishandbuch BPMN*. Hanser Fachbuchverlag, 2010. (Cited on page 19.)

[165] Michael Gaitanides. *Prozessorganisation. Entwicklung, Ansätze und Programme prozessorientierter Organisationsgestaltung*. Vahlen, 1983. (Cited on pages 3, 5, and 49.)

[166] Luciano García-Bañuelos. Pattern identification and classification in the translation from BPMN to BPEL. In Meersman and Tari [307], pages 436–444. ISBN 978-3-540-88870-3. (Cited on pages 27 and 47.)

[167] Javier Andrade Garda, Juan Ares Casal, Rafael García Vázquez, Juan Pazos, Santiago Rodríguez Yánez, and Andrés Silva. A methodological framework for viewpoint-oriented conceptual modeling. *IEEE Trans. Software Eng.*, 30(5):282–294, 2004. (Cited on page 176.)

[168] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Company, 1979. (Cited on page 51.)

[169] Dragan Gasevic, Giancarlo Guizzardi, Kuldar Taveter, and Gerd Wagner. Vocabularies, ontologies, and rules for enterprise and business process modeling and management. *Inf. Syst.*, 35(4):375–378, 2010. (Cited on page 48.)

[170] Michael Gelfond and Jorge Lobo. Authorization and obligation policies in dynamic systems. In Maria Garcia de la Banda and Enrico Pontelli, editors, *ICLP*, volume 5366 of *Lecture Notes in Computer Science*, pages 22–36. Springer, 2008. ISBN 978-3-540-89981-5. (Cited on page 276.)

[171] Christian Gerth, Jochen Malte Küster, and Gregor Engels. Language-independent change management of process models. In Andy Schürr and Bran Selic, editors, *MoDELS*, volume 5795 of *Lecture Notes in Computer Science*, pages 152–166. Springer, 2009. ISBN 978-3-642-04424-3. (Cited on page 212.)

[172] Christian Gierds and Jan Sürmeli, editors. *2nd Central-European Workshop on Services and their Composition, Services und ihre Komposition, ZEUS 2010, Berlin, Germany, February 25-26, 2010. Proceedings*, volume 563 of *CEUR Workshop Proceedings*, 2010. CEUR-WS.org. (Cited on page 323.)

[173] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-match: an algorithm and an implementation of semantic matching. In Christoph Bussler, John Davies, Dieter Fensel, and Rudi Studer, editors, *ESWS*, volume 3053 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2004. ISBN 3-540-21999-4. (Cited on page 50.)

[174] Christian W. Günther. *Process Mining in Flexible Environments*. PhD thesis, Eindhoven University of Technology, 2009. (Cited on pages 264, 265, and 266.)

[175] Stijn Goedertier and Jan Vanthienen. Designing compliant business processes with obligations and permissions. In Eder and Dustdar [139], pages 5–14. ISBN 3-540-38444-8. (Cited on page 42.)

[176] Stijn Goedertier, David Martens, Jan Vanthienen, and Bart Baesens. Robust process discovery with artificial negative events. *Journal of Machine Learning Research*, 10:1305–1340, 2009. (Cited on pages 254 and 275.)

[177] Ursula Goltz and Wolfgang Reisig. The non-sequential behavior of Petri nets. *Information and Control*, 57(2/3):125–147, 1983. (Cited on page 86.)

[178] Jaap Gordijn and Roel Wieringa. A value-oriented approach to e-business process design. In Johann Eder and Michele Missikoff, editors, *CAiSE*, volume 2681 of *Lecture Notes in Computer Science*, pages 390–403. Springer, 2003. ISBN 3-540-40442-2. (Cited on page 49.)

[179] Jaap Gordijn, Eric Yu, and Bas van der Raadt. E-service design using i* and e^{3}value modeling. *IEEE Software*, 23(3):26–33, 2006. (Cited on page 49.)

[180] Florian Gottschalk, Wil M. P. van der Aalst, and Monique H. Jansen-Vullers. Merging event-driven process chains. In Meersman and Tari [307], pages 418–426. ISBN 978-3-540-88870-3. (Cited on pages 218, 244, and 247.)

[181] Florian Gottschalk, Wil M. P. van der Aalst, Monique H. Jansen-Vullers, and Marcello La Rosa. Configurable workflow models. *Int. J. Cooperative Inf. Syst.*, 17(2):177–221, 2008. (Cited on pages 217 and 246.)

[182] Gemma Grau, Xavier Franch, and Neil A. M. Maiden. Prim: An i*-based process reengineering method for information systems specification. *Information & Software Technology*, 50(1-2):76–100, 2008. (Cited on page 49.)

[183] Gianluigi Greco, Antonella Guzzo, Luigi Pontieri, and Domenico Saccà. Mining expressive process models by clustering workflow traces. In Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors, *PAKDD*, volume 3056 of *Lecture Notes in Computer Science*, pages 52–62. Springer, 2004. ISBN 3-540-22064-X. (Cited on pages 254 and 275.)

[184] Daniela Grigori, Juan Carlos Corrales, and Mokrane Bouzeghoub. Behavioral matchmaking for service retrieval: Application to conversation protocols. *Inf. Syst.*, 33(7-8):681–698, 2008. (Cited on pages 50 and 51.)

[185] Martin L. Griss. Implementing product-line features with component reuse. In William B. Frakes, editor, *ICSR*, volume 1844 of *Lecture Notes in Computer Science*, pages 137–152. Springer, 2000. ISBN 3-540-67696-1. (Cited on page 245.)

[186] Varun Grover, Kirk D. Fiedler, and James Teng. Exploring the success of information technology enabled business process reengineering. *IEEE Transactions on Engineering Management*, 41(3):1–8, 1994. (Cited on pages 3, 7, 49, and 163.)

[187] Thomas Gschwind, Jana Koehler, and Janette Wong. Applying patterns during business process modeling. In Dumas et al. [135], pages 4–19. ISBN 978-3-540-85757-0. (Cited on page 145.)

[188] Giancarlo Guizzardi, Heinrich Herre, and Gerd Wagner. On the general ontological foundations of conceptual modeling. In Stefano Spaccapietra, Salvatore T. March, and Yahiko Kambayashi, editors, *ER*, volume 2503 of *Lecture Notes in Computer Science*, pages 65–78. Springer, 2002. ISBN 3-540-44277-4. (Cited on page 48.)

[189] Christian W. Günther and Wil M. P. van der Aalst. Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In Alonso et al. [19], pages 328–343. ISBN 978-3-540-75182-3. (Cited on page 248.)

[190] Carsten Gutwenger and Petra Mutzel. A linear time implementation of SPQR-trees. In Joe Marks, editor, *Graph Drawing*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2000. ISBN 3-540-41554-8. (Cited on pages 109 and 116.)

[191] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. Issues in modeling process variants with Provop. In Ardagna et al. [24], pages 56–67. ISBN 978-3-642-00327-1. (Cited on page 246.)

[192] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. Guaranteeing soundness of configurable process variants in Provop. In Birgit Hofreiter and Hannes Werthner, editors, *CEC*, pages 98–105. IEEE Computer Society, 2009. ISBN 978-0-7695-3755-9. (Cited on page 247.)

[193] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. Capturing variability in business process models: the Provop approach. *Journal of Software Maintenance*, 22(6-7):519–546, 2010. (Cited on page 246.)

[194] Terry A. Halpin and Matthew Curland. Automated verbalization for ORM 2. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *OTM Workshops (2)*, volume 4278 of *Lecture Notes in Computer Science*, pages 1181–1190. Springer, 2006. ISBN 3-540-48273-3. (Cited on page 60.)

[195] Michael Hammer and James Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. HarperBusiness, 1993. (Cited on pages 3, 5, and 49.)

[196] Richard W. Hamming. Error detection and error correction codes. *The Bell System Technical Journal*, XXVI(2):147–160, 1950. (Cited on page 50.)

[197] David Harel and Rami Marelly. Specifying and executing behavioral requirements: the play-in/play-out approach. *Software and System Modeling*, 2(2):82–107, 2003. (Cited on page 144.)

[198] David Harel and Amnon Naamad. The statemate semantics of statecharts. *ACM Trans. Softw. Eng. Methodol.*, 5(4):293–333, 1996. (Cited on page 23.)

[199] David Harel and Bernhard Rumpe. Meaningful modeling: What's the semantics of 'semantics'? *IEEE Computer*, 37(10):64–72, 2004. (Cited on page 17.)

[200] Paul Harmon. *Business Process Change: A Guide for Business Managers and BPM and Six Sigma Professionals*. Morgan Kaufman, 2007. (Cited on page 7.)

[201] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. (Cited on page 51.)

[202] Rainer Hauser and Jana Koehler. Compiling process graphs into executable code. In Gabor Karsai and Eelco Visser, editors, *GPCE*, volume 3286 of *Lecture Notes in Computer Science*, pages 317–336. Springer, 2004. ISBN 3-540-23580-9. (Cited on pages 27 and 47.)

[203] Bin He and Kevin Chen-Chuan Chang. Automatic complex schema matching across web query interfaces: A correlation mining approach. *ACM Trans. Database Syst.*, 31(1):346–395, 2006. (Cited on page 54.)

[204] Keijo Heljanko. Deadlock and reachability checking with finite complete prefixes. Research Report A56, Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory for Theoretical Computer Science, Espoo, Finland, December 1999. (Cited on pages 118 and 140.)

[205] John C. Henderson and N. Venkatraman. Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal*, 38(2/3):472–484, 1999. (Cited on page 42.)

[206] Martin Henkel, Jelena Zdravkovic, and Paul Johannesson. Service-based processes: design for business and technology. In Aiello et al. [15], pages 21–29. ISBN 1-58113-871-7. (Cited on pages 49, 54, 166, 174, and 176.)

[207] Sallie M. Henry and Dennis G. Kafura. Software structure metrics based on information flow. *IEEE Trans. Software Eng.*, 7(5):510–518, 1981. (Cited on page 209.)

[208] Wolfgang Hesse. More matters on (meta-)modelling: remarks on Thomas Kühne's "matters". *Software and System Modeling*, 5(4):387–394, 2006. (Cited on page 4.)

[209] Jan Hidders, Marlon Dumas, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Jan Verelst. When are two workflows the same? In Mike D. Atkinson and Frank K. H. A. Dehne, editors, *CATS*, volume 41 of *CRPIT*, pages 3–11. Australian Computer Society, 2005. ISBN 1-920682-23-6. (Cited on pages 86 and 150.)

[210] Janelle B. Hill, Michele Cantara, Marc Kerremans, and Daryl C. Plummer. Magic Quadrant for Business Process Management Suites. Technical report, Gartner Research, February 2009. (Cited on pages 11 and 18.)

[211] Sebastian Hinz, Karsten Schmidt, and Christian Stahl. Transforming BPEL to Petri nets. In van der Aalst et al. [458], pages 220–235. ISBN 3-540-28238-6. (Cited on pages 25 and 35.)

[212] Tin Kam Ho. Stop word location and identification for adaptive text recognition. *IJDAR*, 3(1):16–26, 2000. (Cited on pages 50 and 60.)

[213] Charles A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978. (Cited on pages 85 and 172.)

[214] Thomas Hornung, Agnes Koschmider, and Georg Lausen. Recommendation based process modeling support: Method and user experience. In Li et al. [277], pages 265–278. ISBN 978-3-540-87876-6. (Cited on pages 50, 51, and 60.)

[215] Ian Horrocks and James A. Hendler, editors. *The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*, volume 2342 of *Lecture Notes in Computer Science*, 2002. Springer. ISBN 3-540-43760-6. (Cited on pages 290 and 312.)

[216] Volker Hoyer, Eva Bucherer, and Florian Schnabel. Collaborative e-business process modelling: Transforming private EPC to public BPMN business process models. In ter Hofstede et al. [438], pages 185–196. ISBN 978-3-540-78237-7. (Cited on pages 22 and 47.)

[217] Richard Hull, Jan Mendling, and Stefan Tai, editors. *Business Process Management - 8th International Conference, BPM 2010, Hoboken, NJ, USA, September 13-16, 2010. Proceedings*, volume 6336 of *Lecture Notes in Computer Science*, 2010. Springer. ISBN 978-3-642-15617-5. (Cited on pages 291, 313, 323, and 324.)

[218] Marta Indulska, Peter F. Green, Jan Recker, and Michael Rosemann. Business process modeling: Perceived benefits. In Laender et al. [260], pages 458–471. ISBN 978-3-642-04839-5. (Cited on page 7.)

[219] Matthew A. Jaro. Advances in record linking methodology as applied to the 1985 census of Tampa Florida. *Journal of the American Statistical Society*, 84(406):414–20, 1989. (Cited on page 50.)

[220] Kurt Jensen. Coloured Petri nets. In Brauer et al. [63], pages 248–299. ISBN 3-540-17905-4. (Cited on pages 35 and 286.)

[221] Kurt Jensen and Wil M. P. van der Aalst, editors. *Transactions on Petri Nets and Other Models of Concurrency II, Special Issue on Concurrency in Process-Aware Information Systems*, volume 5460 of *Lecture Notes in Computer Science*, 2009. Springer. ISBN 978-3-642-00898-6. (Cited on pages 307 and 314.)

[222] Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. Finding similar questions in large question and answer archives. In Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken, editors, *CIKM*, pages 84–90. ACM, 2005. ISBN 1-59593-140-6. (Cited on page 62.)

[223] Richard Johnson, David Pearson, and Keshav Pingali. The program structure tree: Computing control regions in linear time. In *PLDI*, pages 171–185, 1994. (Cited on page 145.)

[224] Yannis Kalfoglou and W. Marco Schorlemmer. Ontology mapping: The state of the art. In Yannis Kalfoglou, W. Marco Schorlemmer, Amit P. Sheth, Steffen Staab, and Michael Uschold, editors, *Semantic Interoperability and Integration*, volume 04391 of *Dagstuhl Seminar Proceedings*. IBFI, Schloss Dagstuhl, Germany, 2005. (Cited on page 42.)

[225] Stephen H. Kan. *Metrics and models in software quality engineering*. Addison-Wesley, 1995. ISBN 978-0-201-63339-9. (Cited on page 209.)

[226] Dimitris Karagiannis, John Mylopoulos, and Margit Schwab. Business process-based regulation compliance: The case of the Sarbanes-Oxley Act. In *RE*, pages 315–321. IEEE, 2007. ISBN 0-7695-2935-6. (Cited on page 42.)

[227] Roland Kaschek. A little theory of abstraction. In Bernhard Rumpe and Wolfgang Hesse, editors, *Modellierung*, volume 45 of *LNI*, pages 75–92. GI, 2004. (Cited on page 4.)

[228] Kathrin Kaschner and Karsten Wolf. Set algebra for service behavior: Applications and constructions. In Dayal et al. [103], pages 193–210. ISBN 978-3-642-03847-1. (Cited on page 245.)

[229] Raman Kazhamiakin, Marco Pistore, and Marco Roveri. A framework for integrating business processes and business requirements. In *EDOC*, pages 9–20. IEEE Computer Society, 2004. ISBN 0-7695-2214-9. (Cited on page 49.)

[230] Raman Kazhamiakin, Marco Pistore, and Luca Santuari. Analysis of communication models in web service compositions. In Carr et al. [84], pages 267–276. ISBN 1-59593-323-9. (Cited on page 177.)

[231] Gerhard Keller, Markus Nüttgens, and August-Wilhelm Scheer. Semantische Prozeßmodellierung auf der Grundlage 'Ereignis-gesteuerter Prozeßketten (EPK)'. Technical Report 89, Veröffentlichungen des Instituts für Wirtschaftsinformatik, Saarbrücken,

1992. URL http://www.uni-saarland.de/fileadmin/user_upload/Fachrichtungen/fr13_BWL/professuren/PDF/heft89.pdf. (Cited on page 20.)

[232] Victor Khomenko, Maciej Koutny, and Alexandre Yakovlev. Logic synthesis for asynchronous circuits based on STG unfoldings and incremental sat. *Fundam. Inform.*, 70(1-2):49–73, 2006. (Cited on page 145.)

[233] Bartek Kiepuszewski, Arthur H. M. ter Hofstede, and Christoph Bussler. On structured workflow modelling. In Benkt Wangler and Lars Bergman, editors, *CAiSE*, volume 1789 of *Lecture Notes in Computer Science*, pages 431–445. Springer, 2000. ISBN 3-540-67630-9. (Cited on page 230.)

[234] Bartek Kiepuszewski, Arthur H. M. ter Hofstede, and Wil M. P. van der Aalst. Fundamentals of control flow in workflows. *Acta Inf.*, 39(3):143–209, 2003. (Cited on pages 86, 94, 96, 97, and 116.)

[235] Won Kim and Jungyun Seo. Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer*, 24(12):12–18, 1991. (Cited on page 47.)

[236] Ekkart Kindler. On the semantics of EPCs: Resolving the vicious circle. *Data Knowl. Eng.*, 56(1):23–40, 2006. (Cited on pages 22, 136, and 138.)

[237] Ralph L. Kliem. Risk management for business process reengineering projects. *IS Management*, 17(4):1–3, 2000. (Cited on page 8.)

[238] Andreas Knöpfel, Bernhard Gröne, and Peter Tabeling. *Fundamental Modeling Concepts*. Wiley, 2005. (Cited on page 177.)

[239] Jana Koehler, Giuliano Tirenni, and Santhosh Kumaran. From business process model to consistent implementation: A case for formal verification methods. In *EDOC*, pages 96–. IEEE Computer Society, 2002. ISBN 0-7695-1742-0. (Cited on page 156.)

[240] Jana Koehler, Rainer Hauser, Shane Sendall, and Michael Wahler. Declarative techniques for model-driven business process integration. *IBM Systems Journal*, 44(1):47–66, 2005. (Cited on pages 27 and 47.)

[241] Jana Koehler, Rainer Hauser, Jochen Malte Küster, Ksenia Ryndina, Jussi Vanhatalo, and Michael Wahler. The role of visual modeling and model transformations in business-driven development. *Electr. Notes Theor. Comput. Sci.*, 211:5–15, 2008. (Cited on page 176.)

[242] George Koliadis, Aleksandar Vranesevic, Moshiur Bhuiyan, Aneesh Krishna, and Aditya K. Ghose. Combining *i** and BPMN for business process model lifecycle management. In Eder and Dustdar [139], pages 416–427. ISBN 3-540-38444-8. (Cited on page 49.)

[243] Alex Kondratyev, Michael Kishinevsky, Alexander Taubin, and Sergei Ten. Analysis of Petri nets by ordering relations in reduced unfoldings. *Formal Methods in System Design*, 12(1):5–38, 1998. (Cited on pages 124, 126, and 134.)

[244] Oliver Kopp, Tobias Unger, and Frank Leymann. Nautilus Event-driven Process Chains: Syntax, semantics, and their mapping to BPEL. In Markus Nüttgens, Frank J. Rump, and Jan Mendling, editors, *EPK*, volume 224 of *CEUR Workshop Proceedings*, pages 85–104. CEUR-WS.org, 2006. (Cited on pages 27 and 47.)

[245] Oliver Kopp, Daniel Martin, Daniel Wutke, and Frank Leymann. The difference between graph-based and block-structured business process modelling languages. *Enterprise Modelling and Information Systems Architectures*, 4(1):3–13, 2009. (Cited on pages 27 and 47.)

[246] Andrei Kovalyov. Concurrency relations and the safety problem for Petri nets. In Kurt Jensen, editor, *Application and Theory of Petri Nets*, volume 616 of *Lecture Notes in Computer Science*, pages 299–309. Springer, 1992. ISBN 3-540-55676-1. (Cited on page 95.)

[247] Andrei Kovalyov and Javier Esparza. A polynomial algorithm to compute the concurrency relation of free-choice signal transition graphs. In R. Smedinga, M.P. Spathopoulos, and P. Kozák, editors, *Proceedings of the International Workshop on Discrete Event Systems (WODES)*, Edinburgh, Scotland, UK, 1996. IEE Society. (Cited on pages 95 and 97.)

[248] John Krogstie, Guttorm Sindre, and Håvard D. Jørgensen. Process models representing knowledge for action: a revised quality framework. *EJIS*, 15(1):91–102, 2006. (Cited on page 50.)

[249] John Krogstie, Andreas L. Opdahl, and Guttorm Sindre, editors. *Advanced Information Systems Engineering, 19th International Conference, CAiSE 2007, Trondheim, Norway, June 11-15, 2007, Proceedings*, volume 4495 of *Lecture Notes in Computer Science*, 2007. Springer. ISBN 978-3-540-72987-7. (Cited on pages 295 and 315.)

[250] Martin Kuhlmann, Dalia Shohat, and Gerhard Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *SACMAT*, pages 179–186. ACM, 2003. ISBN 1-58113-681-1. (Cited on page 276.)

[251] Thomas Kühne. Matters of (meta-)modeling. *Software and System Modeling*, 5(4):369–385, 2006. (Cited on pages 3 and 4.)

[252] Thomas Kühne. Clarifying matters of (meta-) modeling: an author's reply. *Software and System Modeling*, 5(4):395–401, 2006. (Cited on page 4.)

[253] Matthias Kunze, Matthias Weidlich, and Mathias Weske. Behavioral similarity – a proper metric. In *Proceedings of the 9th International Conference on Business Process Management (BPM'11)*, Clermont-Ferrand, France, 2011. To appear. (Cited on pages 214, 282, and 283.)

[254] Matthias Kunze, Matthias Weidlich, and Mathias Weske. m^3 – a behavioral similarity metric for business processes. In Daniel Eichhorn, Agnes Koschmider, and Huayu Zhang, editors, *Proceedings of the 3rd Central-European Workshop on Services and their Composition (ZEUS'11)*, volume 705 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011. (Cited on pages 214 and 282.)

[255] Jochen M. Küster, Christian Gerth, Alexander Förster, and Gregor Engels. Process merging in business-driven development. Technical report, IBM Research Report RZ 3703, IBM Zurich Research Laboratory, 2008. (Cited on page 218.)

[256] Jochen Malte Küster, Christian Gerth, Alexander Förster, and Gregor Engels. Detecting and resolving process model differences in the absence of a change log. In Dumas et al. [135], pages 244–260. ISBN 978-3-540-85757-0. (Cited on pages 52, 145, and 212.)

[257] Jochen Malte Küster, Christian Gerth, and Gregor Engels. Dependent and conflicting change operations of process models. In Richard F. Paige,

Alan Hartman, and Arend Rensink, editors, *ECMDA-FA*, volume 5562 of *Lecture Notes in Computer Science*, pages 158–173. Springer, 2009. ISBN 978-3-642-02673-7. (Cited on page 213.)

[258] Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco M. Dijkman. Merging business process models. In Meersman et al. [309], pages 96–113. ISBN 978-3-642-16933-5. (Cited on pages 51, 218, 244, and 247.)

[259] Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco M. Dijkman. Business process model merging: An approach to business process consolidation. Technical report, Queensland University of Technology, 2010. (Cited on pages 217, 218, 244, and 247.)

[260] Alberto H. F. Laender, Silvana Castano, Umeshwar Dayal, Fabio Casati, and José Palazzo Moreira de Oliveira, editors. *Conceptual Modeling - ER 2009, 28th International Conference on Conceptual Modeling, Gramado, Brazil, November 9-12, 2009. Proceedings*, volume 5829 of *Lecture Notes in Computer Science*, 2009. Springer. ISBN 978-3-642-04839-5. (Cited on pages 289 and 303.)

[261] Alexei Lapouchnian, Yijun Yu, and John Mylopoulos. Requirements-driven design and configuration management of business processes. In Alonso et al. [19], pages 246–261. ISBN 978-3-540-75182-3. (Cited on page 49.)

[262] Antti M. Latva-Koivisto. Finding a complexity measure for business process models. Technical report, Helsinki University of Technology, 2001. (Cited on page 210.)

[263] Ralf Laue and Volker Gruhn. Complexity metrics for business process models. In Abramowicz and Mayr [4], pages 1–12. ISBN 3-88579-179-X. (Cited on page 209.)

[264] Ralf Laue and Jan Mendling. Structuredness and its significance for correctness of process models. *Inf. Syst. E-Business Management*, 8(3): 287–307, 2010. (Cited on page 230.)

[265] Gang Soo Lee and Jung-Mo Yoon. An empirical study on the complexity metrics of Petri nets. *Microelectronics and Reliability*, 32(3):323–329, 1992. (Cited on pages 209 and 210.)

[266] Henrik Leopold, Sergey Smirnov, and Jan Mendling. Refactoring of process model activity labels. In Christina J. Hopfe, Yacine Rezgui, Elisabeth Métais, Alun D. Preece, and Haijiang Li, editors, *NLDB*, volume 6177 of *Lecture Notes in Computer Science*, pages 268–276. Springer, 2010. ISBN 978-3-642-13880-5. (Cited on page 51.)

[267] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966. (Cited on pages 50, 57, and 211.)

[268] Beth Levin. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, Illinois, 1993. (Cited on page 48.)

[269] Frank Leymann. Web Services Flow Language (WSFL 1.0). Technical report, IBM Software Group, May 2001. (Cited on page 25.)

[270] Frank Leymann and Dieter Roller. *Production Work Flow: Concepts and Techniques*. Prentice Hall International, 1999. (Cited on page 7.)

[271] Chen Li, Manfred Reichert, and Andreas Wombacher. On measuring process model similarity based on high-level change operations. In Li et al. [277], pages 248–264. ISBN 978-3-540-87876-6. (Cited on pages 51, 52, 87, 144, 211, 212, and 213.)

[272] Chen Li, Manfred Reichert, and Andreas Wombacher. Discovering reference process models by mining process variants. In *ICWS*, pages 45–53. IEEE Computer Society, 2008. ISBN 978-0-7695-3310-0. (Cited on pages 87, 144, and 247.)

[273] Chen Li, Manfred Reichert, and Andreas Wombacher. Discovering reference models by mining process variants using a heuristic approach. In Dayal et al. [103], pages 344–362. ISBN 978-3-642-03847-1. (Cited on pages 87, 144, and 247.)

[274] Chen Li, Manfred Reichert, and Andreas Wombacher. Representing block-structured process models as order matrices: Basic concepts, formal properties, algorithms. Technical report, University of Twente, Enschede, The Netherlands, 2009. (Cited on pages 87 and 144.)

[275] Chen Li, Manfred Reichert, and Andreas Wombacher. The MinAdept clustering approach for discovering reference process models out of process variants. *Int. J. Cooperative Inf. Syst.*, 19(3-4):159–203, 2010. (Cited on page 247.)

[276] Ninghui Li, Mahesh V. Tripunitara, and Ziad Bizri. On mutually exclusive roles and separation-of-duty. *ACM Trans. Inf. Syst. Secur.*, 10(2), 2007. (Cited on page 276.)

[277] Qing Li, Stefano Spaccapietra, Eric S. K. Yu, and Antoni Olivé, editors. *Conceptual Modeling - ER 2008, 27th International Conference on Conceptual Modeling, Barcelona, Spain, October 20-24, 2008. Proceedings*, volume 5231 of *Lecture Notes in Computer Science*, 2008. Springer. ISBN 978-3-540-87876-6. (Cited on pages 302 and 307.)

[278] Odd Ivar Lindland, Guttorm Sindre, and Arne Sølvberg. Understanding quality in conceptual modeling. *IEEE Software*, 11(2):42–49, 1994. (Cited on pages 4 and 253.)

[279] Richard Lipton. The reachability problem requires exponential space. Technical Report 62, Computer Science Dept., Yale University, 1976. (Cited on page 93.)

[280] Duen-Ren Liu and Minxin Shen. Workflow modeling for virtual processes: an order-preserving process-view approach. *Inf. Syst.*, 28(6):505–532, 2003. (Cited on page 176.)

[281] Niels Lohmann. A feature-complete Petri net semantics for WS-BPEL 2.0. In Marlon Dumas and Reiko Heckel, editors, *WS-FM*, volume 4937 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2007. ISBN 978-3-540-79229-1. (Cited on pages 25, 35, 37, and 39.)

[282] Niels Lohmann, Peter Massuthe, Christian Stahl, and Daniela Weinberg. Analyzing interacting WS-BPEL processes using flexible model generation. *Data Knowl. Eng.*, 64(1):38–54, 2008. (Cited on page 177.)

[283] Niels Lohmann, Eric Verbeek, and Remco M. Dijkman. Petri net transformations for business processes - a survey. In *T. Petri Nets and Other Models of Concurrency* Jensen and van der Aalst [221], pages 46–63. (Cited on pages 35, 36, and 37.)

[284] Edward S. Lowry and C. W. Medlock. Object code optimization. *Commun. ACM*, 12(1):13–22, 1969. (Cited on page 99.)

[285] Ruopeng Lu and Shazia Wasim Sadiq. On the discovery of preferred work practice through business process variants. In Christine Parent, Klaus-Dieter Schewe, Veda C. Storey, and Bernhard Thalheim, editors, *ER*, volume 4801 of *Lecture Notes in Computer Science*, pages 165–180. Springer, 2007. ISBN 978-3-540-75562-3. (Cited on pages 51 and 52.)

[286] Ruopeng Lu, Shazia Wasim Sadiq, and Guido Governatori. Measurement of compliance distance in business processes. *IS Management*, 25 (4):344–355, 2008. (Cited on page 42.)

[287] Ruopeng Lu, Shazia Wasim Sadiq, and Guido Governatori. On managing business processes variants. *Data Knowl. Eng.*, 68(7):642–664, 2009. (Cited on page 245.)

[288] Roberto Lucchi and Manuel Mazzara. A pi-calculus based semantics for WS-BPEL. *J. Log. Algebr. Program.*, 70(1):96–118, 2007. (Cited on pages 25 and 39.)

[289] Jochen Ludewig. Models in software engineering - an introduction. *Inform., Forsch. Entwickl.*, 18(3-4):105–112, 2004. (Cited on page 4.)

[290] Jerry N. Luftman. Assessing IT/business alignment. *IS Management*, 20 (4):9–15, 2003. (Cited on page 42.)

[291] Ken Lunn, Andrew Sixsmith, Ann Lindsay, and Marja Vaarama. Traceability in requirements through process modelling, applied to social care applications. *Information & Software Technology*, 45(15):1045–1052, 2003. (Cited on pages 155 and 172.)

[292] Linh Thao Ly, Stefanie Rinderle-Ma, and Peter Dadam. Design and verification of instantiable compliance rule graphs in process-aware information systems. In Pernici [348], pages 9–23. ISBN 978-3-642-13093-9. (Cited on page 42.)

[293] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with Cupid. In Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao, and Richard T. Snodgrass, editors, *VLDB*, pages 49–58. Morgan Kaufmann, 2001. ISBN 1-55860-804-4. (Cited on page 55.)

[294] Therani Madhusudan, J. Leon Zhao, and Byron Marshall. A case-based reasoning framework for workflow model management. *Data Knowl. Eng.*, 50(1):87–115, 2004. (Cited on pages 50, 51, and 52.)

[295] Paul P. Maglio, Mathias Weske, Jian Yang, and Marcelo Fantinato, editors. *Service-Oriented Computing - 8th International Conference, ICSOC 2010, San Francisco, CA, USA, December 7-10, 2010. Proceedings*, volume 6470 of *Lecture Notes in Computer Science*, 2010. ISBN 978-3-642-17357-8. (Cited on pages 311 and 318.)

[296] Matteo Magnani and Danilo Montesi. BPMN: How much does it cost? An incremental approach. In Alonso et al. [19], pages 80–87. ISBN 978-3-540-75182-3. (Cited on page 7.)

[297] Thomas W. Malone, Kevin Crowston, and George A. Herman, editors. *Organizing Business Knowledge: The MIT Process Handbook*. The MIT Press, Cambridge, Massachusetts, 2003. (Cited on pages 48 and 51.)

[298] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems*. Springer, New York, NY, USA, 1992. ISBN 0-387-97664-7. (Cited on page 88.)

[299] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999. (Cited on page 50.)

[300] Axel Martens. Consistency between executable and abstract processes. In *EEE*, pages 60–67. IEEE Computer Society, 2005. ISBN 0-7695-2073-1. (Cited on page 177.)

[301] Axel Martens. Analyzing web service based business processes. In Maura Cerioli, editor, *FASE*, volume 3442 of *Lecture Notes in Computer Science*, pages 19–33. Springer, 2005. ISBN 3-540-25420-X. (Cited on page 177.)

[302] Antoni Mazurkiewicz. Concurrent program schemes and their interpretations. Technical Report DAIMI Report PB-78, Department of Computer Science, Aarhus University, Denmark, 1977. (Cited on page 32.)

[303] Thomas J. McCabe. A complexity measure. *IEEE Trans. Software Eng.*, 2 (4):308–320, 1976. (Cited on pages 209 and 210.)

[304] Ross M. McConnell and Fabien de Montgolfier. Linear-time modular decomposition of directed graphs. *Discrete Applied Mathematics*, 145:189–209, 2005. (Cited on pages 232, 234, and 238.)

[305] Kenneth L. McMillan. A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1):45–65, 1995. (Cited on pages 118, 119, and 145.)

[306] Robert Meersman and Zahir Tari, editors. *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, OTM Confederated International Conferences, CoopIS, DOA, GADA, and ODBASE 2006, Montpellier, France, October 29 - November 3, 2006. Proceedings, Part I*, volume 4275 of *Lecture Notes in Computer Science*, 2006. Springer. ISBN 3-540-48287-3. (Cited on pages 293, 325, and 326.)

[307] Robert Meersman and Zahir Tari, editors. *On the Move to Meaningful Internet Systems: OTM 2008, OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Proceedings, Part I*, volume 5331 of *Lecture Notes in Computer Science*, 2008. Springer. ISBN 978-3-540-88870-3. (Cited on pages 299, 300, 323, and 327.)

[308] Robert Meersman, Zahir Tari, and Douglas C. Schmidt, editors. *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003*, volume 2888 of *Lecture Notes in Computer Science*, 2003. Springer. ISBN 3-540-20498-9. (Cited on pages 294 and 314.)

[309] Robert Meersman, Tharam S. Dillon, and Pilar Herrero, editors. *On the Move to Meaningful Internet Systems: OTM 2010 - Confederated International Conferences: CoopIS, IS, DOA and ODBASE, Hersonissos, Crete, Greece, October 25-29, 2010, Proceedings, Part I*, volume 6426 of *Lecture Notes in Computer Science*, 2010. Springer. ISBN 978-3-642-16933-5. (Cited on pages 306, 313, and 326.)

[310] Lucas O. Meertens, Maria-Eugenia Iacob, and Silja Eckartz. Feasibility of EPC to BPEL model transformations based on ontology and patterns. In

Rinderle-Ma et al. [385], pages 347–358. ISBN 978-3-642-12185-2. (Cited on pages 27 and 47.)

[311] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128. IEEE Computer Society, 2002. ISBN 0-7695-1531-2. (Cited on page 52.)

[312] Jan Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*, volume 6 of *Lecture Notes in Business Information Processing*. Springer, 2008. ISBN 978-3-540-89223-6. (Cited on pages 17, 22, 135, 174, 209, and 210.)

[313] Jan Mendling and Markus Nüttgens. EPC modelling based on implicit arc types. In Mikhail Godlevsky, Stephen W. Liddle, and Heinrich C. Mayr, editors, *ISTA*, volume 30 of *LNI*, pages 131–142. GI, 2003. ISBN 3-88579-359-8. (Cited on page 22.)

[314] Jan Mendling and Carlo Simon. Business process design by view integration. In Eder and Dustdar [139], pages 55–64. ISBN 3-540-38444-8. (Cited on pages 176, 218, and 244.)

[315] Jan Mendling, Hajo A. Reijers, and Jorge Cardoso. What makes process models understandable? In Alonso et al. [19], pages 48–63. ISBN 978-3-540-75182-3. (Cited on pages 48 and 164.)

[316] Jan Mendling, H. M. W. Verbeek, Boudewijn F. van Dongen, Wil M. P. van der Aalst, and Gustaf Neumann. Detection and prediction of errors in EPCs of the SAP reference model. *Data Knowl. Eng.*, 64(1):312–329, 2008. (Cited on page 136.)

[317] Jan Mendling, Hajo A. Reijers, and Jan Recker. Activity labeling in process modeling: Empirical insights and recommendations. *Inf. Syst.*, 35 (4):467–482, 2010. (Cited on pages 48, 163, and 283.)

[318] Jan Mendling, Hajo A. Reijers, and Wil M. P. van der Aalst. Seven process modeling guidelines (7PMG). *Information & Software Technology*, 52(2):127–136, 2010. (Cited on page 50.)

[319] Christopher Menzel and Michael Grüninger. A formal foundation for process modeling. In *FOIS*, pages 256–269, 2001. (Cited on page 48.)

[320] Derek Miers. Best practice (BPM). *ACM Queue*, 4(2):40–48, 2006. (Cited on pages 155 and 172.)

[321] Lawrence D. Miles. *Techniques of value analysis and engineering*. McGraw-Hill (New York), 1961. (Cited on page 48.)

[322] George A. Miller. WordNet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995. (Cited on pages 50 and 51.)

[323] Robin Milner. *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press, 1999. ISBN 978-0-521-65869-0. (Cited on page 39.)

[324] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, Part I and II. *Inf. Comput.*, 100(1):1–77, 1992. (Cited on page 39.)

[325] Mirjam Minor, Alexander Tartakovski, and Ralph Bergmann. Representation and structure-based similarity assessment for agile workflows. In Rosina Weber and Michael M. Richter, editors, *ICCBR*, volume 4626 of *Lecture Notes in Computer Science*, pages 224–238. Springer, 2007. ISBN 978-3-540-74138-1. (Cited on page 52.)

[326] Ian Molloy, Ninghui Li, Tiancheng Li, Ziqing Mao, Qihua Wang, and Jorge Lobo. Evaluating role mining algorithms. In Barbara Carminati and James Joshi, editors, *SACMAT*, pages 95–104. ACM, 2009. ISBN 978-1-60558-537-6. (Cited on page 276.)

[327] Marco Montali, Maja Pesic, Wil M. P. van der Aalst, Federico Chesani, Paola Mello, and Sergio Storari. Declarative specification and verification of service choreographiess. *TWEB*, 4(1), 2010. (Cited on pages 11, 18, and 88.)

[328] Sandro Morasca. Measuring attributes of concurrent software specifications in Petri nets. In *IEEE METRICS*, pages 100–110. IEEE Computer Society, 1999. ISBN 0-7695-0403-5. (Cited on page 210.)

[329] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989. (Cited on pages 28, 79, 109, 152, 194, 239, and 240.)

[330] Shiva Nejati, Mehrdad Sabetzadeh, Marsha Chechik, Steve M. Easterbrook, and Pamela Zave. Matching and merging of statecharts specifications. In *ICSE*, pages 54–64. IEEE Computer Society, 2007. (Cited on pages 51, 53, 182, 212, 218, 244, and 247.)

[331] Uwe Nestmann and Frank Puhlmann. *Process Algebra for Parallel and Distributed Computing*, chapter Business Process Specification and Analysis, pages 129–160. Boca Raton, Chapmann & Hall/CRC Press, 2009. (Cited on page 39.)

[332] Mark E. Nissen. Valuing IT through virtual process measurement. In *ICIS*, pages 309–323, 1994. (Cited on page 210.)

[333] Natalya Fridman Noy and Michel C. A. Klein. Ontology evolution: Not the same as schema evolution. *Knowl. Inf. Syst.*, 6(4):428–440, 2004. (Cited on page 42.)

[334] Bashar Nuseibeh, Jeff Kramer, and Anthony Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *IEEE Trans. Software Eng.*, 20(10):760–773, 1994. (Cited on page 176.)

[335] Bashar Nuseibeh, Jeff Kramer, and Anthony Finkelstein. Viewpoints: meaningful relationships are difficult! In *ICSE*, pages 676–683. IEEE Computer Society, 2003. (Cited on page 177.)

[336] Markus Nüttgens and Frank J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In Jörg Desel and Mathias Weske, editors, *Promise*, volume 21 of *LNI*, pages 64–77. GI, 2002. ISBN 3-88579-350-4. (Cited on page 20.)

[337] Olivia Oanea, Jan Sürmeli, and Karsten Wolf. Service discovery using communication fingerprints. In Maglio et al. [295], pages 612–618. ISBN 978-3-642-17357-8. (Cited on page 92.)

[338] Sejong Oh, Ravi S. Sandhu, and Xinwen Zhang. An effective role administration model using organization structure. *ACM Trans. Inf. Syst. Secur.*, 9(2):113–137, 2006. (Cited on page 276.)

[339] Chun Ouyang, Marlon Dumas, Stephan Breutel, and Arthur H. M. ter Hofstede. Translating standard process models to BPEL. In Dubois and Pohl [133], pages 417–432. ISBN 3-540-34652-X. (Cited on pages 27 and 47.)

[340] Chun Ouyang, Marlon Dumas, Arthur H. M. ter Hofstede, and Wil M. P. van der Aalst. From BPMN process models to BPEL web services. In *ICWS*, pages 285–292. IEEE Computer Society, 2006. ISBN 0-7695-2669-1. (Cited on pages 27 and 47.)

[341] Chun Ouyang, Eric Verbeek, Wil M. P. van der Aalst, Stephan Breutel, Marlon Dumas, and Arthur H. M. ter Hofstede. Formal semantics and analysis of control flow in WS-BPEL. *Sci. Comput. Program.*, 67(2-3):162–198, 2007. (Cited on pages 25 and 35.)

[342] Susan S. Owicki and Leslie Lamport. Proving liveness properties of concurrent programs. *ACM Trans. Program. Lang. Syst.*, 4(3):455–495, 1982. (Cited on page 79.)

[343] Victor Pankratius and Wolffried Stucky. A formal foundation for workflow composition, workflow view definition, and workflow normalization based on Petri nets. In Sven Hartmann and Markus Stumptner, editors, *APCCM*, volume 43 of *CRPIT*, pages 79–88. Australian Computer Society, 2005. ISBN 1-920682-25-2. (Cited on pages 176, 218, 219, and 244.)

[344] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara. Semantic matching of web services capabilities. In Horrocks and Hendler [215], pages 333–347. ISBN 3-540-43760-6. (Cited on page 51.)

[345] Oscar Pastor and João Falcão e Cunha, editors. *Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005, Proceedings*, volume 3520 of *Lecture Notes in Computer Science*, 2005. Springer. ISBN 3-540-26095-1. (Cited on pages 288 and 316.)

[346] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. WordNet: Similarity - measuring the relatedness of concepts. In Deborah L. McGuinness and George Ferguson, editors, *AAAI*, pages 1024–1025. AAAI Press / The MIT Press, 2004. ISBN 0-262-51183-5. (Cited on page 50.)

[347] Mor Peleg, Iwei Yeh, and Russ B. Altman. Modelling biological processes using workflow and Petri net models. *Bioinformatics*, 18(6):825–837, 2002. (Cited on page 28.)

[348] Barbara Pernici, editor. *Advanced Information Systems Engineering, 22nd International Conference, CAiSE 2010, Hammamet, Tunisia, June 7-9, 2010. Proceedings*, volume 6051 of *Lecture Notes in Computer Science*, 2010. Springer. ISBN 978-3-642-13093-9. (Cited on pages 287, 308, 317, and 324.)

[349] Nicolas Peters and Matthias Weidlich. Using glossaries to enhance the label quality in business process models. In *Proceedings of the 8th GI-Workshop Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (EPK)*, volume 554 of *CEUR*, pages 75–90, Berlin, Germany, November 2009. CEUR-WS.org. (Cited on page 283.)

[350] Nicolas Peters and Matthias Weidlich. Automatic generation of glossaries for process modelling support. *Enterprise Modelling and Information Systems Architectures*, 6(1):30–46, 2011. (Cited on pages 50 and 283.)

[351] James L. Peterson. Petri nets. *ACM Comput. Surv.*, 9(3):223–252, 1977. (Cited on page 152.)

[352] Marian Petre. Why looking isn't always seeing: Readership skills and graphical programming. *Commun. ACM*, 38(6):33–44, 1995. (Cited on page 164.)

[353] Carl Adam Petri. *Kommunikation mit Automaten*. Schriften des IIM Nr. 2, Institut für Instrumentelle Mathematik, Bonn, 1962. (Cited on page 28.)

[354] Carl Adam Petri. Non-sequential processes. Technical Report ISF-77-5, GMD, St-Augustin, Germany, 1977. (Cited on page 86.)

[355] Yves Pigneur and Carson Woo, editors. *Proceedings of the CAISE*06 Workshop on Business/IT Alignment and Interoperability BUSITAL '06, Luxemburg, June 5-9, 2006*, volume 237 of *CEUR Workshop Proceedings*, 2007. CEUR-WS.org. (Cited on pages 288 and 295.)

[356] Klaus Pohl, Günter Böckle, and Frank van der Linden. *Software product line engineering - foundations, principles, and techniques*. Springer, 2005. ISBN 978-3-540-24372-4. (Cited on page 245.)

[357] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. The triconnected abstraction of process models. In Dayal et al. [103], pages 229–244. ISBN 978-3-642-03847-1. (Cited on pages 106, 108, and 145.)

[358] Artem Polyvyanyy, Luciano García-Bañuelos, and Marlon Dumas. Structuring acyclic process models. In Hull et al. [217], pages 276–293. ISBN 978-3-642-15617-5. (Cited on pages 47, 230, 231, 232, and 235.)

[359] Artem Polyvyanyy, Jussi Vanhatalo, and Hagen Völzer. Simplified computation and generalization of the refined process structure tree. In Bravetti and Bultan [65], pages 25–41. ISBN 978-3-642-19588-4. (Cited on pages 60, 104, 106, 108, 109, 116, and 212.)

[360] Artem Polyvyanyy, Matthias Weidlich, and Mathias Weske. The biconnected verification of workflow nets. In Meersman et al. [309], pages 410–418. ISBN 978-3-642-16933-5. (Cited on page 145.)

[361] Artem Polyvyanyy, Luciano García-Bañuelos, and Marlon Dumas. Structuring acyclic process models. *Information Systems*, 2011. To appear. (Cited on pages 22, 37, 47, 136, 143, 145, 230, 231, and 235.)

[362] Artem Polyvyanyy, Luciano García-Bañuelos, Dirk Fahland, and Mathias Weske. Maximal structuring of acyclic process models. Technical report, Hasso Plattner Institute, University of Potsdam, 2011. (Cited on pages 231, 232, and 235.)

[363] Lucia Pomello, Grzegorz Rozenberg, and Carla Simone. A survey of equivalence notions for net based systems. In Grzegorz Rozenberg, editor, *Advances in Petri Nets: The DEMON Project*, volume 609 of *Lecture Notes in Computer Science*, pages 410–472. Springer, 1992. ISBN 3-540-55610-9. (Cited on pages 86 and 150.)

[364] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980. (Cited on pages 50 and 60.)

[365] Michael F. Porter. *Competitive Advantage: Creating and Sustaining Superior Performance*. Free Press, 1998. (Cited on page 49.)

[366] Günter Preuner, Stefan Conrad, and Michael Schrefl. View integration of behavior in object-oriented databases. *Data Knowl. Eng.*, 36(2):153–183, 2001. (Cited on pages 49, 54, 176, 218, and 244.)

[367] Frank Puhlmann and Mathias Weske. Using the pi-calculus for formalizing workflow patterns. In van der Aalst et al. [458], pages 153–168. ISBN 3-540-28238-6. (Cited on page 39.)

[368] Frank Puhlmann and Mathias Weske. A look around the corner: The pi-calculus. In *T. Petri Nets and Other Models of Concurrency* Jensen and van der Aalst [221], pages 64–78. (Cited on page 39.)

[369] Yuzhong Qu, Wei Hu, and Gong Cheng. Constructing virtual documents for ontology matching. In Carr et al. [84], pages 23–31. ISBN 1-59593-323-9. (Cited on page 59.)

[370] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001. (Cited on pages 42, 47, and 54.)

[371] Maryam Razavian and Ramtin Khosravi. Modeling variability in business process models using UML. In *ITNG*, pages 82–87. IEEE Computer Society, 2008. (Cited on page 246.)

[372] Jan Recker and Alexander Dreiling. Does it matter which process modelling language we teach or use? An experimental study on understanding process modelling languages without formal education. In *18th Australasian Conference on Information Systems (ACIS)*, pages 356–366, 2007. (Cited on page 164.)

[373] Jan Recker and Jan Mendling. On the translation between BPMN and BPEL: Conceptual mismatch between process modeling languages. In Thibaud Latour and Michael Petit, editors, *Proceedings 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortiums*, pages 521–532, 2006. (Cited on pages 27 and 47.)

[374] Blaize Horner Reich and Izak Benbasat. Factors that influence the social dimension of alignment between business and information technology objectives. *MIS Quarterly*, 24(1), 2000. (Cited on page 42.)

[375] Manfred Reichert and Peter Dadam. $\text{Adept}_{\text{flex}}$-supporting dynamic changes of workflows without losing control. *J. Intell. Inf. Syst.*, 10(2): 93–129, 1998. (Cited on page 245.)

[376] Manfred Reichert, Stefanie Rinderle, and Peter Dadam. On the common support of workflow type and instance changes under correctness constraints. In Meersman et al. [308], pages 407–425. ISBN 3-540-20498-9. (Cited on pages 213 and 245.)

[377] Hajo A. Reijers. *Design and Control of Workflow Processes: Business Process Management for the Service Industry*, volume 2617 of *Lecture Notes in Computer Science*. Springer, 2003. ISBN 3-540-01186-2. (Cited on page 7.)

[378] Hajo A. Reijers and Irene T. P. Vanderfeesten. Cohesion and coupling metrics for workflow process design. In Jörg Desel, Barbara Pernici, and Mathias Weske, editors, *Business Process Management*, volume 3080 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2004. ISBN 3-540-22235-9. (Cited on page 210.)

[379] Hajo A. Reijers, R. S. Mans, and Robert A. van der Toorn. Improved model management with aggregated business process models. *Data Knowl. Eng.*, 68(2):221–243, 2009. (Cited on pages 218 and 246.)

[380] Wolfgang Reisig. *Petri Nets: An Introduction*, volume 4 of *Monographs in Theoretical Computer Science. An EATCS Series*. Springer, 1985. ISBN 3-540-13723-8. (Cited on page 28.)

[381] Wolfgang Reisig. *Petrinetze: Modellierungstechnik, Analysemethoden, Fallstudien*. Vieweg+Teubner, 2010. ISBN 3-834-81290-0. (Cited on page 28.)

[382] Wolfgang Reisig and Grzegorz Rozenberg, editors. *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, volume 1491 of *Lecture Notes in Computer Science*, 1998. Springer. ISBN 3-540-65306-6. (Cited on pages 28 and 319.)

[383] Stefanie Rinderle, Manfred Reichert, and Peter Dadam. Disjoint and overlapping process changes: Challenges, solutions, applications. In Robert Meersman and Zahir Tari, editors, *CoopIS/DOA/ODBASE (1)*, volume 3290 of *Lecture Notes in Computer Science*, pages 101–120. Springer, 2004. ISBN 3-540-23663-5. (Cited on pages 213 and 245.)

[384] Stefanie Rinderle-Ma, Linh Thao Ly, and Peter Dadam. Business process compliance. *EMISA Forum*, 28(2):24–29, 2008. (Cited on page 8.)

[385] Stefanie Rinderle-Ma, Shazia Wasim Sadiq, and Frank Leymann, editors. *Business Process Management Workshops, BPM 2009 International Workshops, Ulm, Germany, September 7, 2009. Revised Papers*, volume 43 of *Lecture Notes in Business Information Processing*, 2010. Springer. ISBN 978-3-642-12185-2. (Cited on pages 289 and 310.)

[386] Stephen Robertson and Karen S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976. (Cited on pages 50 and 60.)

[387] Andreas Rogge-Solti, Matthias Kunze, Ahmed Awad, and Mathias Weske. Business process configuration wizard and consistency checker for BPMN 2.0. In T. Halpin, S. Nurcan, J. Krogstie, P. Soffer, E. Proper, R. Schmidt, and I. Bider, editors, *Enterprise, Business-Process and Information Systems Modeling. Proceedings of the 12th International Conference BPMDS 2011, and the 16th International Conference EMMSAD 2011.*, volume 81 of *Lecture Notes in Business Information Processing*, 2011. (Cited on page 284.)

[388] Colette Rolland and Naveen Prakash. Bridging the gap between organisational needs and ERP functionality. *Requir. Eng.*, 5(3):180–193, 2000. (Cited on pages 3, 7, 49, and 163.)

[389] Marcello La Rosa, Johannes Lux, Stefan Seidel, Marlon Dumas, and Arthur H. M. ter Hofstede. Questionnaire-driven configuration of reference process models. In Krogstie et al. [249], pages 424–438. ISBN 978-3-540-72987-7. (Cited on page 246.)

[390] Donald J. Rose and Robert Endre Tarjan. Algorithmic aspects of vertex elimination. In *STOC*, pages 245–254. ACM, 1975. (Cited on page 270.)

[391] M. Rosemann. *Process Management: A Guide for the Design of Business Processes*, chapter Preparation of process modeling, pages 41–78. Springer, Berlin, Germany, 2003. (Cited on pages 48 and 283.)

[392] Michael Rosemann and Wil M. P. van der Aalst. A configurable reference modelling language. *Inf. Syst.*, 32(1):1–23, 2007. (Cited on pages 217 and 246.)

[393] Leonid Ya. Rosenblum and Alexandre Yakovlev. Analyzing semantics of concurrent hardware specifications. In *ICPP (3)*, pages 211–218, 1989. (Cited on pages 86, 87, and 144.)

[394] Grzegorz Rozenberg, editor. *Advances in Petri Nets 1990 [10th International Conference on Applications and Theory of Petri Nets, Bonn, Germany, June 1989, Proceedings]*, volume 483 of *Lecture Notes in Computer Science*, 1991. Springer. ISBN 3-540-53863-1. (Cited on pages 291 and 298.)

[395] Anne Rozinat and Wil M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.*, 33(1):64–95, 2008. (Cited on pages 42, 91, 144, 211, 253, 254, and 275.)

[396] Anne Rozinat, Ana Karla Alves de Medeiros, Christian W. Günther, A. J. M. M. Weijters, and Wil M. P. van der Aalst. The need for a process mining evaluation framework in research and practice. In ter Hofstede et al. [438], pages 84–89. ISBN 978-3-540-78237-7. (Cited on page 252.)

[397] Nick Russell, Arthur H. M. ter Hofstede, David Edmond, and Wil M. P. van der Aalst. Workflow data patterns: Identification, representation and tool support. In Delcambre et al. [117], pages 353–368. ISBN 3-540-29389-2. (Cited on page 48.)

[398] Nick Russell, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and David Edmond. Workflow resource patterns: Identification, representation and tool support. In Pastor and e Cunha [345], pages 216–232. ISBN 3-540-26095-1. (Cited on page 48.)

[399] Nick Russell, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. Workflow exception patterns. In Dubois and Pohl [133], pages 288–302. ISBN 3-540-34652-X. (Cited on page 48.)

[400] Shazia W. Sadiq, Wasim Sadiq, and Maria E. Orlowska. Pockets of flexibility in workflow specification. In Hideko S. Kunii, Sushil Jajodia, and Arne Sølvberg, editors, *ER*, volume 2224 of *Lecture Notes in Computer Science*, pages 513–526. Springer, 2001. ISBN 3-540-42866-6. (Cited on page 245.)

[401] Shazia Wasim Sadiq, Guido Governatori, and Kioumars Namiri. Modeling control objectives for business process compliance. In Alonso et al. [19], pages 149–164. ISBN 978-3-540-75182-3. (Cited on page 42.)

[402] Gerard Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975. (Cited on page 50.)

[403] Partha Sampath and Martin Wirsing. Computing the cost of business processes. In Jianhua Yang, Athula Ginige, Heinrich C. Mayr, and Ralf-Detlef Kutsche, editors, *UNISCON*, volume 20 of *Lecture Notes in Business Information Processing*, pages 178–183. Springer, 2009. ISBN 978-3-642-01111-5. (Cited on page 7.)

[404] Davide Sangiorgi and David Walker. *The pi-calculus: A Theory of Mobile Processes*. Cambridge University Press, 2003. (Cited on page 39.)

[405] August-Wilhelm Scheer and Markus Nüttgens. ARIS architecture and reference models for business process management. In van der Aalst et al. [453], pages 376–389. ISBN 3-540-67454-3. (Cited on page 20.)

[406] Hermann J. Schmelzer and Wolfgang Sesselmann. *Geschäftsprozessmanagement in der Praxis*. Hanser Verlag, 2006. (Cited on pages 3, 5, and 49.)

[407] Arnd Schnieders and Frank Puhlmann. Variability mechanisms in e-business process families. In Abramowicz and Mayr [4], pages 583–601. ISBN 3-88579-179-X. (Cited on page 246.)

[408] Pierre-Yves Schobbens, Patrick Heymans, Jean-Christophe Trigaux, and Yves Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479, 2007. (Cited on page 246.)

[409] Andreas Schönberger and Guido Wirtz. Taxonomy on consistency requirements in the business process integration context. In *SEKE*, pages 593–598. Knowledge Systems Institute Graduate School, 2008. ISBN 1-891706-22-5. (Cited on page 177.)

[410] Michael Schrefl and Markus Stumptner. Behavior consistent extension of object life cycles. In Mike P. Papazoglou, editor, *OOER*, volume 1021 of *Lecture Notes in Computer Science*, pages 133–145. Springer, 1995. ISBN 3-540-60672-6. (Cited on pages 151 and 152.)

[411] Michael Schrefl and Markus Stumptner. Behavior consistent refinement of object life cycles. In David W. Embley and Robert C. Goldstein, editors, *ER*, volume 1331 of *Lecture Notes in Computer Science*, pages 155–168. Springer, 1997. ISBN 3-540-63699-4. (Cited on pages 151 and 152.)

[412] Michael Schrefl and Markus Stumptner. Behavior-consistent specialization of object life cycles. *ACM Trans. Softw. Eng. Methodol.*, 11(1):92–148, 2002. (Cited on pages 151, 152, 218, and 244.)

[413] Heiko Schuldt, Gustavo Alonso, Catriel Beeri, and Hans-Jörg Schek. Atomicity and isolation for transactional processes. *ACM Trans. Database Syst.*, 27(1):63–116, 2002. (Cited on page 252.)

[414] John R. Searle. *Speech acts: An essay in the philosophy of language*. Cambridge University Press, Cambridge, 1969. (Cited on page 48.)

[415] Ed Seidewitz. What models mean. *IEEE Software*, 20(5):26–32, 2003. (Cited on page 4.)

[416] Alec Sharp and Patrick McDermott. *Workflow Modeling: Tools for Process Improvement and Application Development*. Artech House, Boston, Massachusetts, 2001. (Cited on page 48.)

[417] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236, 1990. (Cited on page 47.)

[418] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. In Stefano Spaccapietra, editor, *J. Data Semantics IV*, volume 3730 of *Lecture Notes in Computer Science*, pages 146–171. Springer, 2005. ISBN 3-540-31001-0. (Cited on pages 42 and 47.)

[419] Natalia Sidorova, Christian Stahl, and Nikola Trcka. Workflow soundness revisited: Checking correctness in the presence of data while staying conceptual. In Pernici [348], pages 530–544. ISBN 978-3-642-13093-9. (Cited on pages 36 and 286.)

[420] Sase Narine Singh and Carson Woo. Investigating business-IT alignment through multi-disciplinary goal concepts. *Requir. Eng.*, 14(3):177–207, 2009. (Cited on page 42.)

[421] Sergey Smirnov, Matthias Weidlich, Jan Mendling, and Mathias Weske. Action patterns in business process models. In Luciano Baresi, Chi-Hung Chi, and Jun Suzuki, editors, *ICSOC/ServiceWave*, volume 5900 of *Lecture Notes in Computer Science*, pages 115–129, 2009. ISBN 978-3-642-10382-7. (Cited on pages 50 and 283.)

[422] Sergey Smirnov, Remco M. Dijkman, Jan Mendling, and Mathias Weske. Meronymy-based aggregation of activities in business process models. In Jeffrey Parsons, Motoshi Saeki, Peretz Shoval, Carson C. Woo, and Yair Wand, editors, *ER*, volume 6412 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2010. ISBN 978-3-642-16372-2. (Cited on page 51.)

[423] Sergey Smirnov, Matthias Weidlich, and Jan Mendling. Business process model abstraction based on behavioral profiles. In Maglio et al. [295], pages 1–16. ISBN 978-3-642-17357-8. (Cited on page 215.)

[424] Sergey Smirnov, Matthias Weidlich, Jan Mendling, and Mathias Weske. Object-sensitive action patterns in process model repositories. In zur Muehlen and Su [539], pages 251–263. ISBN 978-3-642-20510-1. (Cited on page 283.)

[425] Sergey Smirnov, Matthias Weidlich, and Jan Mendling. Business process model abstraction based on synthesis from well-structured behavioral profiles. Technical report, Hasso Plattner Institute, University of Potsdam, 2011. (Cited on page 215.)

[426] Sergey Smirnov, Matthias Weidlich, Jan Mendling, and Mathias Weske. Action patterns in business process model repositories. Technical report, Hasso Plattner Institute, University of Potsdam, June 2011. (Cited on page 283.)

[427] Oleg Sokolsky, Sampath Kannan, and Insup Lee. Simulation-based graph similarity. In Holger Hermanns and Jens Palsberg, editors, *TACAS*, volume 3920 of *Lecture Notes in Computer Science*, pages 426–440. Springer, 2006. ISBN 3-540-33056-9. (Cited on pages 53, 181, 182, and 212.)

[428] Minseok Song and Wil van der Aalst. Supporting process mining by showing events at a glance. In K. Chari and A. Kumar, editors, *Proceedings of 17th Annual Workshop on Information Technologies and Systems (WITS)*, pages 139–145, Montreal, Canada, 2007. (Cited on page 276.)

[429] Charles Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 100(3/4):441–471, 1987. (Cited on page 169.)

[430] Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer, 1974. (Cited on page 4.)

[431] Tony Spiteri Staines. Intuitive mapping of UML 2 activity diagrams into fundamental modeling concept Petri net diagrams and colored Petri nets. In *ECBS*, pages 191–200. IEEE Computer Society, 2008. (Cited on pages 23 and 35.)

[432] Veda C. Storey. Comparing relationships in conceptual modeling: Mapping to semantic classifications. *IEEE Trans. Knowl. Data Eng.*, 17(11): 1478–1489, 2005. (Cited on page 48.)

[433] Harald Störrle and Jan Hendrik Hausmann. Towards a formal semantics of UML 2.0 activities. In Peter Liggesmeyer, Klaus Pohl, and Michael Goedicke, editors, *Software Engineering*, volume 64 of *LNI*, pages 117–128. GI, 2005. ISBN 3-88579-393-8. (Cited on page 23.)

[434] Mark Strembeck and Gustaf Neumann. An integrated approach to engineer and enforce context constraints in rbac environments. *ACM Trans. Inf. Syst. Secur.*, 7(3):392–427, 2004. (Cited on page 276.)

[435] Jan Sürmeli. Profiling services with static analysis. In *AWPN*, volume 501 of *CEUR Workshop Proceedings*, pages 35–40. CEUR-WS.org, 2009. (Cited on page 92.)

[436] Kaijun Tan, Jason Crampton, and Carl A. Gunter. The consistency of task-based authorization constraints in workflow systems. In *Proceedings of the 17th IEEE Workshop on Computer Security Foundations (CSFW)*, 2004. (Cited on page 276.)

[437] Arthur H. M. ter Hofstede and Henderik Alex Proper. How to formalize it?: Formalization principles for information system development methods. *Information & Software Technology*, 40(10):519–540, 1998. (Cited on page 53.)

[438] Arthur H. M. ter Hofstede, Boualem Benatallah, and Hye-Young Paik, editors. *Business Process Management Workshops, BPM 2007 International Workshops, BPI, BPD, CBP, ProHealth, RefMod, semantics4ws, Brisbane, Australia, September 24, 2007, Revised Selected Papers*, volume 4928 of *Lecture Notes in Computer Science*, 2008. Springer. ISBN 978-3-540-78237-7. (Cited on pages 302 and 316.)

[439] Arthur H. M. ter Hofstede, Wil M. P. van der Aalst, Michael Adams, and Nick Russell, editors. *Modern Business Process Automation: YAWL and its Support Environment*. Springer, 2009. (Cited on pages 7, 36, and 39.)

[440] Satish Thatte. XLANG Web Services for Business Process Design. Technical report, Microsoft Corporation, 2001. (Cited on page 25.)

[441] P. S. Thiagarajan and Klaus Voss. In praise of free choice nets. In Grzegorz Rozenberg, Hartmann J. Genrich, and Gérard Roucairol, editors, *European Workshop on Applications and Theory in Petri Nets*, volume 188 of *Lecture Notes in Computer Science*, pages 438–454. Springer, 1984. ISBN 3-540-15204-0. (Cited on page 33.)

[442] Oliver Thomas and Michael Fellmann. Semantic process modeling - design and implementation of an ontology-based representation of business processes. *Business & Information Systems Engineering*, 1(6):438–451, 2009. (Cited on page 48.)

[443] Sebastián Uchitel and Marsha Chechik. Merging partial behavioural models. In Richard N. Taylor and Matthew B. Dwyer, editors, *SIGSOFT FSE*, pages 43–52. ACM, 2004. ISBN 1-58113-855-5. (Cited on page 218.)

[444] Antti Valmari. The state explosion problem. In Reisig and Rozenberg [382], pages 429–528. ISBN 3-540-65306-6. (Cited on pages 93 and 118.)

[445] Franck van Breugel and Maria Koshika. Models and verification of BPEL. Technical report, York University, 2005. URL `http://www.cse.yorku.ca/~franck/research/drafts/tutorial.pdf`. (Cited on page 25.)

[446] Wil van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011. (Cited on pages 88, 89, 144, 154, 155, 211, 248, 255, and 275.)

[447] Wil M. P. van der Aalst. The application of Petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998. (Cited on pages 33 and 94.)

[448] Wil M. P. van der Aalst. Workflow verification: Finding control-flow errors using Petri-net-based techniques. In van der Aalst et al. [453], pages 161–183. ISBN 3-540-67454-3. (Cited on pages 35, 94, and 175.)

[449] Wil M. P. van der Aalst. Inheritance of business processes: A journey visiting four notorious problems. In Hartmut Ehrig, Wolfgang Reisig, Grzegorz Rozenberg, and Herbert Weber, editors, *Petri Net Technology for Communication-Based Systems*, volume 2472 of *Lecture Notes in Computer Science*, pages 383–408. Springer, 2003. ISBN 3-540-20538-1. (Cited on pages 238 and 240.)

[450] Wil M. P. van der Aalst. Business alignment: using process mining as a tool for delta analysis and conformance testing. *Requir. Eng.*, 10(3): 198–211, 2005. (Cited on page 42.)

[451] Wil M. P. van der Aalst and Twan Basten. Life-cycle inheritance: A Petri-net-based approach. In Pierre Azéma and Gianfranco Balbo, editors, *ICATPN*, volume 1248 of *Lecture Notes in Computer Science*, pages 62–81. Springer, 1997. ISBN 3-540-63139-9. (Cited on pages 35, 152, 172, 191, 192, 218, 244, and 286.)

[452] Wil M. P. van der Aalst and Maja Pesic. Decserflow: Towards a truly declarative service flow language. In Bravetti et al. [66], pages 1–23. ISBN 3-540-38862-1. (Cited on pages 11, 18, 79, and 88.)

[453] Wil M. P. van der Aalst, Jörg Desel, and Andreas Oberweis, editors. *Business Process Management, Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, 2000. Springer. ISBN 3-540-67454-3. (Cited on pages 289, 316, and 320.)

[454] Wil M. P. van der Aalst, Paulo Barthelmess, Clarence A. Ellis, and Jacques Wainer. Proclets: A framework for lightweight interacting workflow processes. *Int. J. Cooperative Inf. Syst.*, 10(4):443–481, 2001. (Cited on page 245.)

[455] Wil M. P. van der Aalst, Alexander Hirnschall, and H. M. W. (Eric) Verbeek. An alternative way to analyze workflow graphs. In Anne Banks Pidduck, John Mylopoulos, Carson C. Woo, and M. Tamer Özsu, editors, *CAiSE*, volume 2348 of *Lecture Notes in Computer Science*, pages 535–552. Springer, 2002. ISBN 3-540-43738-X. (Cited on page 108.)

[456] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003. (Cited on pages 20, 48, 87, and 283.)

[457] Wil M. P. van der Aalst, A.J.M.M. Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004. (Cited on pages 88, 144, 154, 211, 248, and 275.)

[458] Wil M. P. van der Aalst, Boualem Benatallah, Fabio Casati, and Francisco Curbera, editors. *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649, 2005. ISBN 3-540-28238-6. (Cited on pages 302 and 314.)

[459] Wil M. P. van der Aalst, Ana Karla A. de Medeiros, and A. J. M. M. Weijters. Genetic process mining. In Gianfranco Ciardo and Philippe Darondeau, editors, *ICATPN*, volume 3536 of *Lecture Notes in Computer Science*, pages 48–69. Springer, 2005. ISBN 3-540-26301-2. (Cited on pages 90, 144, 211, 248, and 250.)

[460] Wil M. P. van der Aalst, Hajo A. Reijers, A. J. M. M. Weijters, Boudewijn F. van Dongen, Ana Karla Alves de Medeiros, Minseok Song, and H. M. W. (Eric) Verbeek. Business process mining: An industrial application. *Inf. Syst.*, 32(5):713–732, 2007. (Cited on pages 88, 248, 253, and 275.)

[461] Wil M. P. van der Aalst, Marlon Dumas, Florian Gottschalk, Arthur H. M. ter Hofstede, Marcello La Rosa, and Jan Mendling. Correctness-preserving configuration of business process models. In José Luiz Fiadeiro and Paola Inverardi, editors, *FASE*, volume 4961 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2008. ISBN 978-3-540-78742-6. (Cited on pages 246 and 247.)

[462] Wil M. P. van der Aalst, Maja Pesic, and Helen Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science - R&D*, 23(2):99–113, 2009. (Cited on pages 11, 18, 79, and 88.)

[463] Wil M. P. van der Aalst, Boudewijn F. van Dongen, Christian W. Günther, Anne Rozinat, Eric Verbeek, and A.J.M.M. Weijters. ProM: The process mining toolkit. In Ana Karla A. de Medeiros and Barbara Weber, editors, *BPM (Demos)*, volume 489 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009. (Cited on page 272.)

[464] Wil M. P. van der Aalst, Marlon Dumas, Florian Gottschalk, Arthur H. M. ter Hofstede, Marcello La Rosa, and Jan Mendling. Preserving correctness during business process model configuration. *Formal Asp. Comput.*, 22(3-4):459–482, 2010. (Cited on page 247.)

[465] Wil M. P. van der Aalst, Niels Lohmann, Peter Massuthe, Christian Stahl, and Karsten Wolf. Multiparty contracts: Agreeing and implementing interorganizational processes. *Comput. J.*, 53(1):90–106, 2010. (Cited on page 177.)

[466] Wil M. P. van der Aalst, Vladimir Rubin, H. M. W. Verbeek, Boudewijn F. van Dongen, Ekkart Kindler, and Christian W. Günther. Process mining: a two-step approach to balance between underfitting and overfitting. *Software and System Modeling*, 9(1):87–111, 2010. (Cited on pages 249 and 253.)

[467] Wil M. P. van der Aalst, Kees M. van Hee, Arthur H. M. ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Asp. Comput.*, 23(3):333–363, 2011. (Cited on page 35.)

[468] Wil M.P. van der Aalst. Pi-calculus versus Petri nets: Let us eat humble pie rather than further inflate the pi-hype. *BPTrends*, 3(5):1–11, May 2005. (Cited on page 39.)

[469] Jan Martijn E. M. van der Werf, Boudewijn F. van Dongen, Cor A. J. Hurkens, and Alexander Serebrenik. Process discovery using integer linear programming. *Fundam. Inform.*, 94(3-4):387–412, 2009. (Cited on page 249.)

[470] Boudewijn F. van Dongen, Jan Mendling, and Wil M. P. van der Aalst. Structural patterns for soundness of business process models. In *EDOC*, pages 116–128. IEEE Computer Society, 2006. ISBN 0-7695-2558-X. (Cited on pages 91, 145, and 212.)

[471] Boudewijn F. van Dongen, Nadia Busi, G. Michele Pinna, and Wil M.P. van der Aalst. An iterative algorithm for applying the theory of regions in process mining. In Wolfgang Reisig, Kees van Hee, and Karsten Wolf,

editors, *Proceedings of the Workshop on Formal Approaches to Business Processes and Web Services (FABPWS'07)*, Siedlce, Poland, 2007. Publishing House of University of Podlasie. (Cited on page 249.)

[472] Boudewijn F. van Dongen, Monique H. Jansen-Vullers, H. M. W. Verbeek, and Wil M. P. van der Aalst. Verification of the SAP reference models using EPC reduction, state-space analysis, and invariants. *Computers in Industry*, 58(6):578–601, 2007. (Cited on pages 22, 35, 37, 38, and 136.)

[473] Boudewijn F. van Dongen, Remco M. Dijkman, and Jan Mendling. Measuring similarity between business process models. In Bellahsene and Léonard [42], pages 450–464. ISBN 978-3-540-69533-2. (Cited on pages 50, 51, 52, 53, 91, 145, and 212.)

[474] Rob J. van Glabbeek. The linear time-branching time spectrum (extended abstract). In Jos C. M. Baeten and Jan Willem Klop, editors, *CONCUR*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer, 1990. ISBN 3-540-53048-7. (Cited on pages 85, 86, and 150.)

[475] Rob J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions*. PhD thesis, Free University, Amsterdam, 1990. Introduction available at `http://theory.stanford.edu/~rvg/thesis.html`. Second edition available as *CWI tract* 109, CWI, Amsterdam 1996. (Cited on page 12.)

[476] Rob J. van Glabbeek. The linear time - branching time spectrum ii. In Eike Best, editor, *CONCUR*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 1993. ISBN 3-540-57208-2. (Cited on pages 85 and 150.)

[477] Rob J. van Glabbeek and Ursula Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Inf.*, 37(4/5):229–327, 2001. (Cited on page 151.)

[478] Rob J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *J. ACM*, 43(3):555–600, 1996. (Cited on pages 85 and 177.)

[479] Jussi Vanhatalo, Hagen Völzer, and Frank Leymann. Faster and more focused control-flow analysis for business process models through SESE decomposition. In Bernd J. Krämer, Kwei-Jay Lin, and Priya Narasimhan, editors, *ICSOC*, volume 4749 of *Lecture Notes in Computer Science*, pages 43–55. Springer, 2007. ISBN 978-3-540-74973-8. (Cited on page 145.)

[480] Jussi Vanhatalo, Hagen Völzer, Frank Leymann, and Simon Moser. Automatic workflow graph refactoring and completion. In Bouguettaya et al. [60], pages 100–115. ISBN 978-3-540-89647-0. (Cited on pages 36 and 145.)

[481] Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. The refined process structure tree. *Data Knowl. Eng.*, 68(9):793–818, 2009. (Cited on pages 60, 104, 109, 145, and 212.)

[482] Anisha Vemulapalli and Nary Subramanian. Transforming functional requirements from UML into BPEL to efficiently develop SOA-based systems. In Robert Meersman, Pilar Herrero, and Tharam S. Dillon, editors, *OTM Workshops*, volume 5872 of *Lecture Notes in Computer Science*, pages 337–349. Springer, 2009. ISBN 978-3-642-05289-7. (Cited on pages 27 and 47.)

[483] Pepijn Visser, Dean Jones, Trevor Bench-Capon, and Michael Shave. Assessing heterogeneity by classifying ontology mismatches. In *Proc. 1st International Conference on Formal Ontology in Information Systems (FOIS)*, pages 148–162, Trento (IT), 1998. (Cited on page 47.)

[484] Hagen Völzer. A new semantics for the inclusive converging gateway in safe processes. In Hull et al. [217], pages 294–309. ISBN 978-3-642-15617-5. (Cited on page 19.)

[485] Yair Wand and Ron Weber. Research commentary: Information systems and conceptual modeling - a research agenda. *Information Systems Research*, 13(4):363–376, 2002. (Cited on page 17.)

[486] Stephen Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, 1962. (Cited on pages 97 and 126.)

[487] Carolyn R. Watters. Information retrieval and the virtual document. *JASIS*, 50(11):1028–1029, 1999. (Cited on page 59.)

[488] Barbara Weber, Manfred Reichert, and Stefanie Rinderle-Ma. Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data Knowl. Eng.*, 66(3):438–466, 2008. (Cited on pages 49, 54, 201, 213, 245, and 246.)

[489] Barbara Weber, Shazia Wasim Sadiq, and Manfred Reichert. Beyond rigidity - dynamic process lifecycle support. *Computer Science - R&D*, 23 (2):47–65, 2009. (Cited on page 245.)

[490] Ingo Weber, Jörg Hoffmann, and Jan Mendling. Beyond soundness: on the verification of semantic business process models. *Distributed and Parallel Databases*, 27(3):271–343, 2010. (Cited on page 48.)

[491] Jonathan J. Webster and Chunyu Kit. Tokenization as the initial phase in NLP. In *COLING*, pages 1106–1110, 1992. (Cited on page 50.)

[492] Jochen De Weerdt, Manu De Backer, Jan Vanthienen, and Bart Baesens. A critical evaluation study of model-log metrics in process discovery. In zur Muehlen and Su [539], pages 158–169. ISBN 978-3-642-20510-1. (Cited on page 252.)

[493] Matthias Weidlich and Jan Mendling. Perceived consistency between process models. *Inf. Syst.*, 2011. URL `http://dx.doi.org/10.1016/j.is.2010.12.004`. To appear. (Cited on pages 149 and 166.)

[494] Matthias Weidlich and Mathias Weske. Structural and behavioural commonalities of process variants. In Gierds and Sürmeli [172], pages 41–48. (Cited on pages 51 and 215.)

[495] Matthias Weidlich and Mathias Weske. On the behavioural dimension of correspondences between process models. In Gierds and Sürmeli [172], pages 65–72. (Cited on page 41.)

[496] Matthias Weidlich, Gero Decker, and Mathias Weske. Efficient analysis of BPEL 2.0 processes using pi-calculus. In Jie Li, Minyi Guo, Qun Jin, Yongbing Zhang, Liang-Jie Zhang, Hai Jin, Masahiro Mambo, Jiro Tanaka, and Hiromu Hayashi, editors, *APSCC*, pages 266–274. IEEE, 2007. ISBN 0-7695-3051-6. (Cited on pages 25 and 39.)

[497] Matthias Weidlich, Gero Decker, Alexander Großkopf, and Mathias Weske. BPEL to BPMN: The myth of a straight-forward mapping. In Meersman and Tari [307], pages 265–282. ISBN 978-3-540-88870-3. (Cited on pages 27 and 47.)

[498] Matthias Weidlich, Alistair P. Barros, Jan Mendling, and Mathias Weske. Vertical alignment of process models - how can we get there? In Terry A. Halpin, John Krogstie, Selmin Nurcan, Erik Proper, Rainer Schmidt, Pnina Soffer, and Roland Ukor, editors, *BMMDS/EMMSAD*, volume 29 of *Lecture Notes in Business Information Processing*, pages 71–84. Springer, 2009. ISBN 978-3-642-01861-9. (Cited on pages 41, 49, 54, 166, and 174.)

[499] Matthias Weidlich, Mathias Weske, and Jan Mendling. Change propagation in process models using behavioural profiles. In *IEEE SCC*, pages 33–40. IEEE Computer Society, 2009. (Cited on page 179.)

[500] Matthias Weidlich, Remco M. Dijkman, and Jan Mendling. The ICoP framework: Identification of correspondences between process models. In Pernici [348], pages 483–498. ISBN 978-3-642-13093-9. (Cited on page 41.)

[501] Matthias Weidlich, Remco M. Dijkman, and Mathias Weske. Deciding behaviour compatibility of complex correspondences between process models. In Hull et al. [217], pages 78–94. ISBN 978-3-642-15617-5. (Cited on pages 153 and 172.)

[502] Matthias Weidlich, Felix Elliger, and Mathias Weske. Generalised computation of behavioural profiles based on Petri-net unfoldings. In Bravetti and Bultan [65], pages 101–115. ISBN 978-3-642-19588-4. (Cited on page 93.)

[503] Matthias Weidlich, Artem Polyvyanyy, Nirmit Desai, and Jan Mendling. Process compliance measurement based on behavioural profiles. In Pernici [348], pages 499–514. ISBN 978-3-642-13093-9. (Cited on page 251.)

[504] Matthias Weidlich, Artem Polyvyanyy, Jan Mendling, and Mathias Weske. Efficient computation of causal behavioural profiles using structural decomposition. In Johan Lilius and Wojciech Penczek, editors, *Petri Nets*, volume 6128 of *Lecture Notes in Computer Science*, pages 63–83. Springer, 2010. ISBN 978-3-642-13674-0. (Cited on pages 73 and 93.)

[505] Matthias Weidlich, Jan Mendling, and Mathias Weske. Efficient consistency measurement based on behavioral profiles of process models. *IEEE Trans. Software Eng.*, 37(3):410–429, 2011. (Cited on pages 73, 93, 149, and 179.)

[506] Matthias Weidlich, Jan Mendling, and Mathias Weske. A foundational approach for managing process variability. In *Proceedings of the 23rd International Conference on Advanced Information Systems Engineering (CAiSE'11)*, volume 6741 of *Lecture Notes in Computer Science*, pages 267–282. Springer, 2011. (Cited on page 215.)

[507] Matthias Weidlich, Artem Polyvyanyy, Nirmit Desai, Jan Mendling, and Mathias Weske. Process compliance analysis based on behavioural profiles. *Inf. Syst.*, 36(7):1009–1025, 2011. (Cited on page 251.)

[508] Matthias Weidlich, Artem Polyvyanyy, Jan Mendling, and Mathias Weske. Causal behavioural profiles - efficient computation, applications, and evaluation. *Fundam. Inform.*, 2011. To appear. (Cited on pages 73 and 93.)

[509] Matthias Weidlich, Holger Ziekow, Jan Mendling, Oliver Günther, Mathias Weske, and Nirmit Desai. Event-based monitoring of process execution violations. In *Proceedings of the 9th International Conference on Business Process Management (BPM'11)*, 2011. To appear. (Cited on page 284.)

[510] A. J. M. M. Weijters and Wil M. P. van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003. (Cited on pages 248 and 264.)

[511] A. J. M. M. Weijters, Wil M. P. van der Aalst, and Ana Karla Alves de Medeiros. Process mining with the HeuristicsMiner algorithm. BETA Working Paper Series WP 166, Eindhoven University of Technology, 2006. (Cited on pages 248, 254, 264, and 275.)

[512] Lijie Wen, Wil M. P. van der Aalst, Jianmin Wang, and Jiaguang Sun. Mining process models with non-free-choice constructs. *Data Min. Knowl. Discov.*, 15(2):145–180, 2007. (Cited on pages 90 and 248.)

[513] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007. ISBN 978-3-540-73521-2. (Cited on pages 3, 5, 6, 17, and 19.)

[514] Stephen A. White. Using BPMN to model a BPEL process:. Technical report, BPTrends, March 2005. (Cited on page 27.)

[515] Stephen A. White and Derek Miers. *BPMN Modeling and Reference Guide: Understanding and Using BPMN*. Future Strategies Inc, 2008. (Cited on page 19.)

[516] Roel Wieringa and Jaap Gordijn. Value-oriented design of service coordination processes: correctness and trust. In Hisham Haddad, Lorie M. Liebrock, Andrea Omicini, and Roger L. Wainwright, editors, *SAC*, pages 1320–1327. ACM, 2005. ISBN 1-58113-964-0. (Cited on page 49.)

[517] William E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1990. (Cited on page 50.)

[518] Petia Wohed, Wil M. P. van der Aalst, Marlon Dumas, and Arthur H. M. ter Hofstede. Analysis of web services composition languages: The case of BPEL4WS. In Il-Yeol Song, Stephen W. Liddle, Tok Wang Ling, and Peter Scheuermann, editors, *ER*, volume 2813 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2003. ISBN 3-540-20299-4. (Cited on page 48.)

[519] Petia Wohed, Wil M. P. van der Aalst, Marlon Dumas, Arthur H. M. ter Hofstede, and Nick Russell. Pattern-based analysis of the control-flow perspective of UML activity diagrams. In Delcambre et al. [117], pages 63–78. ISBN 3-540-29389-2. (Cited on page 48.)

[520] Petia Wohed, Wil M. P. van der Aalst, Marlon Dumas, Arthur H. M. ter Hofstede, and Nick Russell. On the suitability of BPMN for business process modelling. In Schahram Dustdar, José Luiz Fiadeiro, and Amit P. Sheth, editors, *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 161–176. Springer, 2006. ISBN 3-540-38901-6. (Cited on page 48.)

[521] Christian Wolter and Andreas Schaad. Modeling of task-based authorization constraints in BPMN. In Alonso et al. [19], pages 64–79. ISBN 978-3-540-75182-3. (Cited on page 276.)

[522] Andreas Wombacher. Evaluation of technical measures for workflow similarity based on a pilot study. In Meersman and Tari [306], pages 255–272. ISBN 3-540-48287-3. (Cited on pages 53 and 211.)

[523] Andreas Wombacher and Chen Li. Alternative approaches for workflow similarity. In *IEEE SCC*, pages 337–345. IEEE Computer Society, 2010. ISBN 978-0-7695-4126-6. (Cited on pages 53 and 211.)

[524] Andreas Wombacher and Maarten Rozie. Piloting an empirical study on measures for workflow similarity. In *IEEE SCC*, pages 94–102. IEEE Computer Society, 2006. ISBN 0-7695-2670-5. (Cited on pages 53 and 211.)

[525] Andreas Wombacher and Maarten Rozie. Evaluation of workflow similarity measures in service discovery. In Mareike Schoop, Christian Huemer, Michael Rebstock, and Martin Bichler, editors, *Service Oriented Electronic Commerce*, volume 80 of *LNI*, pages 51–71. GI, 2006. ISBN 3-88579-174-9. (Cited on pages 50, 53, and 211.)

[526] Peter Y. H. Wong and Jeremy Gibbons. A process semantics for BPMN. In Shaoying Liu, T. S. E. Maibaum, and Keijiro Araki, editors, *ICFEM*, volume 5256 of *Lecture Notes in Computer Science*, pages 355–374. Springer, 2008. ISBN 978-3-540-88193-3. (Cited on page 19.)

[527] Martin R. Woodward, Michael A. Hennell, and David Hedley. A measure of control flow complexity in program text. *IEEE Trans. Software Eng.*, 5(1):45–50, 1979. (Cited on page 209.)

[528] Jialiang Wu and Eberhard Voit. Hybrid modeling in biochemical systems theory by means of functional Petri nets. *J. Bioinformatics and Computational Biology*, 7(1):107–134, 2009. (Cited on page 28.)

[529] Li Xu and David W. Embley. Discovering direct and indirect matches for schema elements. In *DASFAA*, pages 39–46. IEEE Computer Society, 2003. (Cited on page 54.)

[530] Zhiqiang Yan, Remco M. Dijkman, and Paul Grefen. Fast business process similarity search with feature-based similarity estimation. In Meersman et al. [309], pages 60–77. ISBN 978-3-642-16933-5. (Cited on pages 50 and 52.)

[531] Johannes Maria Zaha, Alistair P. Barros, Marlon Dumas, and Arthur H. M. ter Hofstede. Let's dance: A language for service behavior modeling. In Meersman and Tari [306], pages 145–162. ISBN 3-540-48287-3. (Cited on page 88.)

[532] Johannes Maria Zaha, Marlon Dumas, Arthur H. M. ter Hofstede, Alistair P. Barros, and Gero Decker. Bridging global and local models of service-oriented systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 38(3):302–318, 2008. (Cited on pages 88 and 177.)

[533] Stephan Zelewski. Petrinetzbasierte Modellierung komplexer Produktionssysteme - Eine Untersuchung des Beitrags von Petrinetzen zur Prozeßkoordinierung in komplexen Produktionssystemen, insbesondere Flexiblen Fertigungssystemen, Band 9: Beurteilung des Petrinetz-Konzepts. Technical report, University of Leipzig, 1995. (in German). (Cited on pages 11 and 150.)

[534] Man Zhang and Zhenhua Duan. From business process models to web services orchestration: The case of UML 2.0 activity diagram to BPEL. In Bouguettaya et al. [60], pages 505–510. ISBN 978-3-540-89647-0. (Cited on pages 27 and 47.)

[535] Xiangpeng Zhao, Hongli Yang, and Zongyan Qiu. Towards the formal model and verification of web service choreography description language. In Bravetti et al. [66], pages 273–287. ISBN 3-540-38862-1. (Cited on page 177.)

[536] Xiaohui Zhao, Chengfei Liu, Wasim Sadiq, and Marek Kowalkiewicz. Process view derivation and composition in a dynamic collaboration environment. In Meersman and Tari [307], pages 82–99. ISBN 978-3-540-88870-3. (Cited on page 176.)

[537] Michael zur Muehlen and Danny Ting-Yi Ho. Risk management in the BPM lifecycle. In Christoph Bussler and Armin Haller, editors, *Business Process Management Workshops*, volume 3812, pages 454–466, 2005. ISBN 3-540-32595-6. (Cited on page 8.)

[538] Michael zur Muehlen and Jan Recker. How much language is enough? Theoretical and practical use of the business process modeling notation. In Bellahsene and Léonard [42], pages 465–479. ISBN 978-3-540-69533-2. (Cited on pages 39 and 165.)

[539] Michael zur Muehlen and Jianwen Su, editors. *Business Process Management Workshops - BPM 2010 International Workshops and Education Track, Hoboken, NJ, USA, September 13-15, 2010, Revised Selected Papers*, volume 66 of *Lecture Notes in Business Information Processing*, 2011. Springer. ISBN 978-3-642-20510-1. (Cited on pages 287, 318, and 323.)

All links were last followed on June 06, 2011.