Temporal propositions as regular languages

Tim Fernando

Trinity College Dublin

Abstract. Temporal propositions are mapped to sets of strings that witness (in a precise sense) the propositions over discrete linear Kripke frames. The strings are collected into regular languages to ensure the decidability of entailments given by inclusions between languages. (Various notions of bounded entailment are shown to be expressible as language inclusions.) The languages unwind computations implicit in the logical (and temporal) connectives via a system of finite-state constraints adapted from finite-state morphology. Applications to Hybrid Logic and non-monotonic inertial reasoning are briefly considered.

1 Introduction

Model-theoretic semantics cashes out the meaning of a formula φ by specifying when a model M satisfies φ. The present work converts a temporal formula φ to a regular language that indicates (in a precise sense) which models M satisfy φ. The strings in these languages are not only easier to grasp computationally than the models M but are natural candidates for events witnessing φ. The languages are formed by adapting finite-state methods widely used in morphology [1].

An instructive example of a temporal formula that can be converted to a regular language is p *until* q, which is equivalent to the disjunction

$$q \ \vee \ (p \wedge next(q)) \ \vee \ (p \wedge next(p \wedge next(q))) \ \vee \ \cdots \tag{1}$$

over discrete linear orders. If we draw the disjuncts as $\boxed{q}$, $\boxed{p}\boxed{q}$, $\boxed{p}\boxed{p}\boxed{q}$ and so on, and rewrite $\vee$ as non-deterministic choice $|$, then (1) becomes the language

$$\boxed{q} \mid \boxed{p}\boxed{q} \mid \boxed{p}\boxed{p}\boxed{q} \mid \ \cdots = \boxed{p}^*\boxed{q}$$

where $\cdot^*$ is Kleene star (for zero or more iterations). In general, given a finite set Φ of formulas and a string $a_1 a_2 \cdots a_n \in Pow(\Phi)^*$ of subsets a_i of Φ, let $fmla_0(a_1 a_2 \cdots a_n)$ be the conjunction

$$fmla_0(a_1 a_2 \cdots a_n) \stackrel{\text{def}}{=} (\bigwedge a_1) \wedge next(\bigwedge a_2) \wedge \cdots \wedge next^{n-1}(\bigwedge a_n)$$

with conjuncts $next^{i-1}(\bigwedge a_i)$ that are themselves built from conjunctions $\bigwedge a_i$. The conjunction of the empty set is, as usual, some fixed tautology $\top$. For instance, we have

$$fmla_0(\boxed{p}\Box\boxed{q,r}) = p \wedge next(next(q \wedge r))$$

where formulas constituting a symbol a_i in a string are enclosed by a box rather than by curly braces $\{\cdot\}$, and the tautology $\top$ for the conjunction $\bigwedge\Box$ of the empty set $\Box$ is suppressed. Also, $\mathit{fmla}_0(\boxed{p}^0\boxed{q}) = q$ and

$$\mathit{fmla}_0(\boxed{p}^{n+1}\boxed{q}) = p \wedge \cdots \wedge \mathit{next}^n(p) \wedge \mathit{next}^{n+1}(q) \ ,$$

making p *until* q equivalent to the disjunction

$$\bigvee \{\mathit{fmla}_0(s) : s \in \boxed{p}^*\boxed{q}\}$$

over the models of interest.

1.1 Representations over the integers

Before specifying what "the models of interest" are, let us not forget past operators such as the converse *prev* of *next*, and sharpen the map fmla_0 accordingly. To mark out the present, we introduce a fresh formula $\mathit{now} \notin \Phi$, refining our picture $\boxed{p}\boxed{q}$ for $p \wedge \mathit{next}(q)$ to $\boxed{\mathit{now}, p}\boxed{q}$, so as to represent $\mathit{prev}(p) \wedge q$ as $\boxed{p}\boxed{\mathit{now}, q}$. Given a subset a of Φ, let us write $a_\dagger$ for the union $a \cup \boxed{\mathit{now}}$, drawing a box instead of $\{\cdot\}$ as we shall form strings from such sets. Let us call a string now_Φ*-pointed* if it has the form $sa_\dagger s'$ for some strings s and s' over the alphabet $\mathit{Pow}(\Phi)$ and some subset a of Φ. We define a backward version $\mathit{almf}(s)$ of $\mathit{fmla}_0(s)$

$$\mathit{almf}(a_1 a_2 \cdots a_n) \stackrel{\text{def}}{=} \mathit{prev}^n(\bigwedge a_1) \wedge \mathit{prev}^{n-1}(\bigwedge a_2) \wedge \cdots \wedge \mathit{prev}(\bigwedge a_n)$$

and map a now_Φ-pointed string $sa_\dagger s'$ to the conjunction

$$\mathit{fmla}(sa_\dagger s') \stackrel{\text{def}}{=} \mathit{almf}(s) \ \wedge \ \mathit{fmla}_0(as') \ .$$

For example, $\mathit{almf}(\boxed{p}\Box) = \mathit{prev}(\mathit{prev}(p))$ and thus,

$$\mathit{fmla}(\boxed{p}\Box\boxed{\mathit{now}, q, r}) = \mathit{prev}(\mathit{prev}(p)) \wedge q \wedge r$$

(dropping $\top$ as before).

Turning now to models, we base our Kripke models for a set P of atomic formulas on not only the natural numbers (for future operators) but also the negative integers (for the past). Let $\mathbb{Z} = \{0, 1, -1, 2, -2, \ldots\}$ be the set of integers, and *satisfaction* $\models$ be defined relative to an integer $x \in \mathbb{Z}$ and a function $V : P \to \mathit{Pow}(\mathbb{Z})$, called a valuation, as follows. For an atomic formula $p \in P$, we set

$$\langle V, x\rangle \models p \iff_{\text{def}} x \in V(p) \ .$$

The unary connective *next* moves us to the successor $x + 1$

$$\langle V, x\rangle \models \mathit{next}(\varphi) \iff_{\text{def}} \langle V, x+1\rangle \models \varphi$$

while *prev* steps back to the preceding integer $x-1$

$$\langle V,x\rangle \models \mathit{prev}(\varphi) \stackrel{\text{def}}{\iff} \langle V,x-1\rangle \models \varphi \ .$$

Conjunctions are formed from the binary connective $\wedge$

$$\langle V,x\rangle \models \varphi\wedge\psi \stackrel{\text{def}}{\iff} \langle V,x\rangle \models \varphi \text{ and } \langle V,x\rangle \models \psi$$

as well as the 0-ary connective $\top$

$$\langle V,x\rangle \models \top \ .$$

In what follows, we let Φ denote some fixed *finite* set of formulas on which $\models$ is well-defined, and form now_Φ-pointed strings s over the alphabet $\mathit{Pow}(\Phi\cup\{\mathit{now}\})$ with formulas $\mathit{fmla}(s)$ on which $\models$ is well-defined (under the clauses above for *next*, *prev*, $\wedge$ and $\top$). We leave the full specification of clauses for $\models$ open-ended, introducing clauses such as

$$\begin{aligned}\langle V,x\rangle \models \varphi \ \mathit{until}\ \psi \stackrel{\text{def}}{\iff} (\exists y\geq x)\ &\langle V,y\rangle \models \psi \text{ and}\\ &(\forall z<y)\ z\geq x \text{ implies } \langle V,z\rangle \models \varphi\end{aligned}$$

to pose the problem of representing a formula such as $\mathit{prev}(p \ \mathit{until}\ (q\wedge r))$.

Definition. A set L of now_Φ-pointed strings *stringwise* Φ*-represents* φ if φ is equivalent to the disjunction $\bigvee\{\mathit{fmla}(s) : s\in L\}$ in that

$$\langle V,x\rangle \models \varphi \iff (\exists s\in L)\ \langle V,x\rangle \models \mathit{fmla}(s)$$

for all $V : P\to \mathit{Pow}(\mathbb{Z})$ and $x\in\mathbb{Z}$.

It is easy to see that $\mathit{prev}(p \ \mathit{until}\ (q\wedge r))$ is stringwise Φ-represented by

$$\boxed{q,r}\boxed{\mathit{now}} \mid \boxed{p}\boxed{\mathit{now},q,r} \mid \boxed{p}\boxed{\mathit{now},p}\boxed{p}^*\boxed{q,r}$$

assuming $p,q,r\in\Phi$. Indeed, every $\varphi\in\Phi$ is stringwise Φ-represented by $\boxed{\mathit{now},\varphi}$. For $\varphi\notin\Phi$, the idea is to turn $\boxed{\mathit{now},\varphi}$ into an "equivalent" set of now_Φ-pointed strings. But some cases are hopeless.

Consider, for instance, the formula $\mathsf{G}\varphi$ asserting that φ is true now and at every point in the future

$$\langle V,x\rangle \models \mathsf{G}\varphi \stackrel{\text{def}}{\iff} (\forall y\geq x)\ \langle V,y\rangle \models \varphi \ .$$

If $\Phi\subseteq P$ is a set of atomic formulas and $p\in P$, then $\mathsf{G}p$ has no stringwise Φ-representation. But we will (in section 4 below) modify the notion of representation so that

$$\Box^*\boxed{\mathit{now},p}\boxed{p}^* \ \mathit{path}\text{wise } \{p\}\text{-represents } \mathsf{G}p$$

and in general, for every set L of now_{Φ}-pointed strings,

$$L \text{ stringwise } \Phi\text{-represents } \varphi \text{ implies } \Box^* L \Box^* \text{ pathwise } \Phi\text{-represents } \varphi.$$

Stringwise or pathwise, can we ensure that our Φ-representations are regular languages? In view of the prevalence of automata-theoretic methods in temporal logic [2, 3], it is perhaps not surprising how much we can. Nonetheless, some delicacy is required to keep these languages regular. For instance, a straightforward analog (within the present setting) of the replace operator in [1] takes us outside the realm of finite automata (see §3.2 below). Instead, we adapt Koskenniemi's restriction operator [1] over an alphabet of symbols that have structure reflecting concurrent computation.

But why should it matter that the languages are regular? One of many useful properties of regular languages is the decidability of inclusions $\subseteq$ between them (as opposed say, to context-free languages). In the present context, entailments are naturally expressed as inclusions between languages (§2.2 below). The regularity of these languages gives us a computational handle on entailments that arguably compensates for the loss of first-order logic due to infinitary disjunctions from Kleene star.[1]

1.2 Related work

Inclusions between languages figure prominently in *Model Checking*, where a system $\mathcal{A}$ satisfies specification $\mathcal{S}$ precisely if $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{S})$ for certain languages $\mathcal{L}(\mathcal{A})$ and $\mathcal{L}(\mathcal{S})$ associated with $\mathcal{A}$ and $\mathcal{S}$, respectively ([3], page 124). The languages in this case consist of infinite strings accepted according to Büchi's criterion. That criterion is closely related to the notion of pathwise Φ-representation spelled out in §4.2 below.

Staying with finite strings, the present approach uses inclusions to define entailments relative to constraints derived from those in [1] through an operation called superposition [4], reviewed in §2.1 below. Superposition supports a form of "true" concurrency different from the non-deterministic interleaving typically associated with temporal logic ([2], page 1017). The notions of entailment induced by superposition do *not* in general preserve length, and therefore cannot be regular under the conventions of *Regular Model Checking* [5]. They are, however, computable by finite-state transducers with componentwise ϵ-moves, and are therefore regular according to [1]. A notable difference between [5] and [1] is that regular relations are closed under intersection in the former but not in the latter. Henceforth, we adopt the latter definition of a regular relation (relaxing the requirement of length preservation). What is important for our purposes is that given a relation R and a language L, the *L-restriction R_L of R* defined by

$$R_L \stackrel{\text{def}}{=} \{(s, s') \in R \ : \ s' \in L\}$$

[1] A routine compactness argument shows that the disjunction (1) above cannot be expressed as a first-order theory.

and its domain, the *Peirce product* $\langle R\rangle L$ *of* R *with* L

$$\langle R\rangle L \stackrel{\text{def}}{=} \{s \ : \ (\exists s')\ sR_Ls'\}\ ,$$

are both regular if R and L are.[2] As explained in §3.1 below, Koskenniemi's restriction operation can be derived from the Peirce product. The transitive closure of a regular relation, length preserving or not, need not be regular. An example in the present context is provided by the obvious analog of the replace operation from [1]. Because its transitive closure need not be regular, we resort instead to Koskenniemi-esque constraints.

2 Inclusions comparing information content

This section shows how to compare the information content of languages through inclusion $\subseteq$. The first way, through subsumption $\trianglerighteq$, is described most directly via an operation of superposition. The second way, through weak subsumption $\underline{\blacktriangleright}$, allows for $\Box$-padding in $\trianglerighteq$. As relations on strings, both $\trianglerighteq$ and $\underline{\blacktriangleright}$ are regular, and can be lifted to languages through Peirce products, leading to natural notions of bounded entailment.

2.1 Superposition and subsumption

Given languages L and L' over the alphabet $Pow(\Phi \cup \{now\})$, the *superposition* $L\&L'$ *of* L *and* L' consists of the componentwise union of strings in L and L' of the same length

$$L\&L' \stackrel{\text{def}}{=} \bigcup_{n\geq 0}\{(a_1 \cup a'_1)\cdots(a_n \cup a'_n)\ :\ a_1\cdots a_n \in L \text{ and } a'_1\cdots a'_n \in L'\}$$

[4]. For instance, the superposition of $\boxed{p}^*\boxed{q}$ and $\boxed{now}\Box^*$ injects *now* into the first box of every string in $\boxed{p}^*\boxed{q}$

$$\boxed{p}^*\boxed{q}\ \&\ \boxed{now}\Box^* = \boxed{now, q}\ |\ \boxed{now, p}\boxed{p}^*\boxed{q}\ .$$

[2] The terminology "Peirce product" is from [6], but the notation $\langle R\rangle L$ is borrowed from dynamic logic [7]. A finite-state transducer for R with transitions $\rightarrow_R$ and a finite automata for L with transitions $\rightarrow_L$ combine to form a finite-state transducer for R_L with transitions

$$(q, q') \stackrel{a,b}{\rightarrow} (r, r') \iff^{\text{def}} q \stackrel{a,b}{\rightarrow}_R r \text{ and } (q' \stackrel{b}{\rightarrow}_L r' \text{ or } (b = \epsilon \text{ and } q' = r'))$$

(as in the usual construction for the intersection of regular languages, but with ϵ-moves). For the Peirce product $\langle R\rangle L$, we existentially quantify b out for

$$(q, q') \stackrel{a}{\rightarrow} (r, r') \iff^{\text{def}} (\exists b)\ q \stackrel{a,b}{\rightarrow}_R r \text{ and } (q' \stackrel{b}{\rightarrow}_L r' \text{ or } (b = \epsilon \text{ and } q' = r'))\ .$$

To accept $L\&L'$ given finite automata A and A' accepting L and L' respectively, we run A and A' in lockstep. That is, an automaton accepting $L\&L'$ can be built as in the usual product construction for the intersection $L \cap L'$ except that the transitions $\rightsquigarrow$ for $L\&L'$ are obtained by unioning the labels on the transitions $\rightarrow_A$ of A and $\rightarrow_{A'}$ of A'

$$(q, q') \overset{b}{\rightsquigarrow} (r, r') \overset{\text{def}}{\iff} (\exists a, a' \subseteq b)\ b = a \cup a' \text{ and } q \overset{a}{\rightarrow}_A r \text{ and } q' \overset{a'}{\rightarrow}_{A'} r' \ .$$

The definition of superposition depends on the assumption that our alphabet consists of sets closed under union. Instead of using subsets of $\Phi \cup \{now\}$ as symbols in our alphabet, we can add a fresh symbol $\wr \notin \Phi \cup \{now\}$, pronounced "tick" (as in a ticking clock), and rewrite, for instance, the string $\boxed{now, p}\boxed{q}$ of length 2 to the string $now\ p\ \wr\ q\ \wr$ of length 5 over the alphabet $\{p, q, now, \wr\}$. This is essentially the approach pursued in [8], taking us back to the usual interleaving model of concurrency except that the passage of time is marked by a tick $\wr$. As noted in [9], one can build finite-state transducers between $Pow(\Phi \cup \{now\})^*$ and $(\Phi \cup \{now, \wr\})^*$ that translate between these in the obvious way and preserve regularity. For convenience, we work with the alphabet $Pow(\Phi \cup \{now\})$, abbreviating it to Σ when the exact choice of Φ is immaterial.

Superposition allows us to compare the information content of languages L and L' over Σ as follows. We say L *subsumes* L' and write $L \trianglerighteq L'$ if L is included in the superposition $L\&L'$

$$L \trianglerighteq L' \overset{\text{def}}{\iff} L \subseteq L\&L' \ .$$

Conflating a string s with the language $\{s\}$, it follows that for strings $a_1 \cdots a_n$ and $b_1 \cdots b_m$ over Σ, $\trianglerighteq$ picks out pairs with the same length $n = m$ and are componentwise related by the converse of $\subseteq$

$$a_1 \cdots a_n \trianglerighteq b_1 \cdots b_m \iff n = m \text{ and } a_i \supseteq b_i \text{ for } 1 \leq i \leq n \ .$$

As a relation on strings, $\trianglerighteq$ is regular; it is computable by a finite-state transducer with one state q_0, both initial and final, and transitions

$$q_0 \overset{a,b}{\rightarrow} q_0 \quad \overset{\text{def}}{\iff} \quad b \subseteq a$$

for all $b, a \in \Sigma$. Taking the Peirce product of $\trianglerighteq$ with a language L, we get

$$s \in \langle \trianglerighteq \rangle L \iff (\exists s' \in L)\ s \trianglerighteq s' \ .$$

We can then restate $L \trianglerighteq L'$ as an inclusion involving the Peirce product $\langle \trianglerighteq \rangle L'$

$$L \trianglerighteq L' \iff L \subseteq \langle \trianglerighteq \rangle L' \ .$$

2.2 Padding and entailments

Although the relation $\trianglerighteq$ will prove useful for formulating constraints later on, it will also be convenient to weaken it slightly so that strings of different length can

be compared. Towards this end, we define for every string $s \in \Sigma^*$ its unpadded form, $unpad(s)$, obtained by deleting all initial and final $\Box$'s from s. That is,

$$unpad(s) \stackrel{\text{def}}{=} \begin{cases} s & \text{if } s \text{ neither begins nor ends with } \Box \\ unpad(s') & \text{if } s = \Box s' \text{ or else if } s = s'\Box \end{cases}$$

so that for example, $unpad(\Box\boxed{p}\Box\boxed{now}\Box\Box) = \boxed{p}\Box\boxed{now}$. As a relation between strings, *unpad* is obviously regular (so long as we don't require length preservation). Next, we say that a string s *weakly subsumes* s' and write $s \underline{\blacktriangleright} s'$ if s subsumes some string equivalent to s' up to unpadding

$$s \underline{\blacktriangleright} s' \stackrel{\text{def}}{\iff} (\exists s'')\ s \trianglerighteq s'' \text{ and } unpad(s'') = unpad(s')\ .$$

It is easy to see that the relation of *unpad*-equivalence

$$\{(s, s') : unpad(s) = unpad(s')\}$$

is regular, making weak subsumption $\underline{\blacktriangleright}$ regular (since regular relations are closed under relational composition).

If we think of strings in a language as possibilities in the same way that worlds in a proposition are under possible worlds semantics (or models of a sentence are in model-theoretic semantics), then it is natural to lift $\underline{\blacktriangleright}$ to sets L, L' of strings through the Peirce product

$$\begin{aligned} L \underline{\blacktriangleright} L' &\stackrel{\text{def}}{\iff} L \subseteq \langle \underline{\blacktriangleright} \rangle L' \\ &\iff (\forall s \in L)(\exists s' \in L')\ s \underline{\blacktriangleright} s' \end{aligned}$$

(paralleling the definition in possible worlds semantics that a proposition p entails p' if $p \subseteq p'$). Defining $L'_\Box$ to be the set of strings *unpad*-equivalent to strings in L'

$$\begin{aligned} L'_\Box &\stackrel{\text{def}}{=} \{s \in \Sigma^* \ :\ (\exists s' \in L')\ unpad(s) = unpad(s')\} \\ &= \Box^* unpad(L') \Box^* \end{aligned}$$

(where $unpad(L') \stackrel{\text{def}}{=} \{unpad(s) : s \in L'\}$), we can relate $\underline{\blacktriangleright}$ back to superposition & via subsumption $\trianglerighteq$ and (un)padding

$$\begin{aligned} L \underline{\blacktriangleright} L' &\iff L \trianglerighteq L'_\Box \\ &\iff L \subseteq L \& L'_\Box\ . \end{aligned}$$

As some strings may represent spurious possibilities, we can weed out strings from $\langle \underline{\blacktriangleright} \rangle L$ by intersecting it with a language C to form

$$\begin{aligned} C[L] &\stackrel{\text{def}}{=} C \cap \langle \underline{\blacktriangleright} \rangle L \\ &= \{s \in C \ :\ s \underline{\blacktriangleright} L\} \end{aligned}$$

which is a regular language whenever L and C are. Recall that regular languages are closed under Boolean operations, including complementation

$$\overline{L} \stackrel{\text{def}}{=} \Sigma^* - L \ .$$

We can express the set of now_{Φ}-pointed strings as $C'[L']$ if we choose C' and L' as follows. Let $L' \stackrel{\text{def}}{=} \boxed{now}$ and let C' be the set of strings that do *not* contain two occurrences of *now*

$$\begin{aligned} C' &\stackrel{\text{def}}{=} \overline{\langle \underline{\blacktriangleright} \rangle (\boxed{now} \Box^* \boxed{now})} \\ &= [\underline{\blacktriangleright}] \overline{\boxed{now} \Box^* \boxed{now}} \end{aligned}$$

where $[R]L$ is the dual of the Peirce product $\langle R \rangle L$

$$[R]L \stackrel{\text{def}}{=} \overline{\langle R \rangle \overline{L}}$$

just as $\forall$ is the dual of $\exists$.

In general, we can beef up $L \underline{\blacktriangleright} L'$ to an entailment $L \vdash_C L'$ by relativizing it to a language C that turns L to $C[L]$

$$\begin{aligned} L \vdash_C L' &\stackrel{\text{def}}{\iff} C[L] \ \underline{\blacktriangleright} \ L' \\ &\iff C \cap \langle \underline{\blacktriangleright} \rangle L \ \subseteq \ \langle \underline{\blacktriangleright} \rangle L' \ . \end{aligned}$$

Clearly, $L \vdash_C L'$ whenever $L \underline{\blacktriangleright} L'$. The introduction of C allows us not only to enlarge a string in L, but also to restrict attention to strings meeting the membership conditions for C

$$L \vdash_C L' \iff (\forall s \in C) \ s \underline{\blacktriangleright} L \text{ implies } s \underline{\blacktriangleright} L' \ .$$

These membership conditions can be viewed as constraints (to satisfy), as we see next.

3 Constraints and their application

In this section, we formulate constraints corresponding to the semantic clauses for $\vee, \wedge, \mathit{next}, \mathit{prev}, \mathit{until}$ and *since*, and apply them to build stringwise representations. For this, a useful regular relation between strings is that of a factor: s' is a *factor of* s if $s = us'v$ for some (possibly empty) strings u and v.

3.1 Constraints conditioned by subsumption

Given languages L and L' over Σ, let $L \Rightarrow L'$ be the set of strings s such that every factor of s that subsumes L also subsumes L'

$$\begin{aligned} L \Rightarrow L' \stackrel{\text{def}}{=} \{ s \in \Sigma^* \ : \ &\text{for every factor } s' \text{ of } s, \\ &s' \trianglerighteq L \text{ implies } s' \trianglerighteq L' \} \ . \end{aligned}$$

For example, to pick out strings that contain $\varphi \wedge \psi$ only if they contain φ and ψ in the same box , we let

$$\boxed{\varphi \wedge \psi} \Rightarrow \boxed{\varphi, \psi} \tag{2}$$

and for disjunction $\varphi \vee \psi$,

$$\boxed{\varphi \vee \psi} \Rightarrow \boxed{\varphi} \mid \boxed{\psi} \; . \tag{3}$$

Writing $\sqsupseteq^L$ for the $\langle \trianglerighteq \rangle L$-restriction of the factor relation

$$s \sqsupseteq^L s' \stackrel{\text{def}}{\iff} s' \text{ is a factor of } s \text{ and } s' \in \langle \trianglerighteq \rangle L \; ,$$

it follows that

$$L \Rightarrow L' = [\sqsupseteq^L] \langle \trianglerighteq \rangle L' \; .$$

As the factor relation is regular, so is $\sqsupseteq^L$ for regular languages L. Thus, since the Peirce product of a regular relation with a regular language is regular, $L \Rightarrow L'$ is a regular language if L and L' are. Indeed,

$$L \Rightarrow L' = \overline{\Sigma^* \; (\langle \trianglerighteq \rangle L \cap \overline{\langle \trianglerighteq \rangle L'}) \; \Sigma^*} \; .$$

Next, we strengthen the constraint $\boxed{next(\varphi)}\Box \Rightarrow \Box\boxed{\varphi}$ to

$$\boxed{next(\varphi)} \stackrel{\mathsf{a}}{\Rightarrow} \boxed{\varphi} \tag{4}$$

where $L \stackrel{\mathsf{a}}{\Rightarrow} L'$ is pronounced "L' after every L" and

$$s \in L \stackrel{\mathsf{a}}{\Rightarrow} L' \stackrel{\text{def}}{\iff} \begin{array}{l} \text{after every factor of } s \text{ that subsumes } L \\ \qquad \text{is a substring that subsumes } L' \end{array}$$

for every string $s \in \Sigma^*$. Defining

$$s \; after^L \; s' \stackrel{\text{def}}{\iff} (\exists u \trianglerighteq \Box^* L) \; s = us' \; ,$$

we get

$$L \stackrel{\mathsf{a}}{\Rightarrow} L' = [after^L] \; \langle \trianglerighteq \rangle (L' \Box^*) \; .$$

It is not difficult to convert a finite automaton for L into a finite-state transducer for $after_L$. Hence, $L \stackrel{\mathsf{a}}{\Rightarrow} L'$ is regular if L and L' are. In fact,

$$L \stackrel{\mathsf{a}}{\Rightarrow} L' = \overline{\langle \trianglerighteq \rangle (\Box^* L) \; \overline{\langle \trianglerighteq \rangle (L' \Box^*)}} \; .$$

Modulo subsumption $\trianglerighteq$, $L \stackrel{\mathsf{a}}{\Rightarrow} L'$ is one form of Koskenniemi's restrictions [1], a second one being $L \stackrel{\mathsf{b}}{\Rightarrow} L'$, read "$L'$ before every L," defined by

$$L \stackrel{\mathsf{b}}{\Rightarrow} L' \stackrel{\text{def}}{=} [before^L] \; \langle \trianglerighteq \rangle (\Box^* L')$$

where

$$s \; before^L \; s' \stackrel{\text{def}}{\iff} (\exists v \trianglerighteq L\Box^*) \; s = s'v \; .$$

As with $\Rightarrow$ and $\stackrel{\mathsf{a}}{\Rightarrow}$, $L \stackrel{\mathsf{b}}{\Rightarrow} L'$ is regular if L and L' are, with

$$L \stackrel{\mathsf{b}}{\Rightarrow} L' = \overline{\overline{\langle\trianglerighteq\rangle(\Box^* L')} \; \langle\trianglerighteq\rangle(L\Box^*)} \; .$$

We also strengthen $\Box\boxed{prev(\varphi)} \Rightarrow \boxed{\varphi}\Box$ to

$$\boxed{prev(\varphi)} \stackrel{\mathsf{b}}{\Rightarrow} \boxed{\varphi} \; . \tag{5}$$

Both $\Rightarrow$ and $\stackrel{\mathsf{a}}{\Rightarrow}$ are used to analyze *until* through auxiliary formulas $\varphi \; ntil \; \psi$

$$\boxed{\varphi \; until \; \psi} \Rightarrow \boxed{\psi} \mid \boxed{\varphi, \; \varphi \; ntil \; \psi} \tag{6}$$

with

$$\boxed{\varphi \; ntil \; \psi} \stackrel{\mathsf{a}}{\Rightarrow} \boxed{\varphi}^* \boxed{\psi} \; . \tag{7}$$

We can treat *since* similarly, using $\stackrel{\mathsf{b}}{\Rightarrow}$ and auxiliary formulas $\varphi \; sinc \; \psi$

$$\boxed{\varphi \; since \; \psi} \Rightarrow \boxed{\psi} \mid \boxed{\varphi, \; \varphi \; sinc \; \psi} \tag{8}$$

$$\boxed{\varphi \; sinc \; \psi} \stackrel{\mathsf{b}}{\Rightarrow} \boxed{\psi}\boxed{\varphi}^* \; . \tag{9}$$

3.2 Application with minimization and projection

Let $P_\bullet$ be the set of formulas constructed from P using $\top, \wedge, \vee, next, prev, until, since, ntil$ and $sinc$. We define a function $\mathcal{C} : P_\bullet \to Pow(P_\bullet \cup \{now\})^*$ mapping a formula $\varphi \in P_\bullet$ to a language $\mathcal{C}(\varphi)$ over the alphabet $Pow(P_\bullet \cup \{now\})$ by induction on φ, using the constraints we have associated above with the connectives

$$\begin{aligned}
\mathcal{C}(\varphi) &\stackrel{\text{def}}{=} [\blacktriangleright]\overline{\boxed{now}\Box^*\boxed{now}} \quad \text{for } \varphi \in P \cup \{\top\} \\
\mathcal{C}(\varphi \wedge \psi) &\stackrel{\text{def}}{=} \mathcal{C}(\varphi) \cap \mathcal{C}(\psi) \cap (\boxed{\varphi \wedge \psi} \Rightarrow \boxed{\varphi, \psi}) \\
\mathcal{C}(\varphi \vee \psi) &\stackrel{\text{def}}{=} \mathcal{C}(\varphi) \cap \mathcal{C}(\psi) \cap (\boxed{\varphi \vee \psi} \Rightarrow \boxed{\varphi} \| \boxed{\psi}) \\
\mathcal{C}(next(\varphi)) &\stackrel{\text{def}}{=} \mathcal{C}(\varphi) \cap (\boxed{next(\varphi)} \stackrel{\mathsf{a}}{\Rightarrow} \boxed{\varphi}) \\
\mathcal{C}(prev(\varphi)) &\stackrel{\text{def}}{=} \mathcal{C}(\varphi) \cap (\boxed{prev(\varphi)} \stackrel{\mathsf{b}}{\Rightarrow} \boxed{\varphi}) \\
\mathcal{C}(\varphi \; until \; \psi) &\stackrel{\text{def}}{=} \mathcal{C}(\varphi \; ntil \; \psi) \cap (\boxed{\varphi \; until \; \psi} \Rightarrow \boxed{\psi} \| \boxed{\varphi, \; \varphi \; ntil \; \psi}) \\
\mathcal{C}(\varphi \; ntil \; \psi) &\stackrel{\text{def}}{=} \mathcal{C}(\varphi) \cap \mathcal{C}(\psi) \cap (\boxed{\varphi \; ntil \; \psi} \stackrel{\mathsf{a}}{\Rightarrow} \boxed{\varphi}^* \boxed{\psi})
\end{aligned}$$

and similarly for *since* and *sinc*. For each $\varphi \in P_\bullet$, the language $\mathcal{C}(\varphi)$ is regular, as is the language

$$\mathcal{C}(\varphi) \cap \langle \blacktriangleright \rangle \boxed{now, \varphi} = \{s \in \mathcal{C}(\varphi) : s \blacktriangleright \boxed{now, \varphi}\}$$

which we shall abbreviate $\hat{\mathcal{C}}(\varphi)$. The language $\hat{\mathcal{C}}(\varphi)$ can be quite massive, but we can reduce it a few ways. The first is through $\trianglerighteq$-minimization: given a language L, define the set $L_{\trianglerighteq}$ of $\trianglerighteq$-*minimal strings in* L by

$$L_{\trianglerighteq} \stackrel{\text{def}}{=} L - \langle \triangleright \rangle L$$

where $\triangleright$ is $\trianglerighteq$ minus equality

$$s \triangleright s' \stackrel{\text{def}}{\iff} s \trianglerighteq s' \text{ and } s \neq s' \ .$$

For example, $(\Box^*|L)_{\trianglerighteq} = \Box^*$. Notice that $L_{\trianglerighteq}$ is regular if L is. The second way of trimming a language is by projecting every string $a_1 \cdots a_n$ in it to the string

$$\rho(a_1 \cdots a_n) \stackrel{\text{def}}{=} (a_1 \cap (P \cup \boxed{now})) \cdots (a_n \cap (P \cup \boxed{now}))$$

restricting the symbols to subsets of $P \cup \boxed{now}$. For instance, if $p \in P$ then $\rho(\boxed{p, \psi \vee \varphi}\boxed{now, prev(\chi)}) = \boxed{p}\boxed{now}$. In general, if L is a regular language, then so is $\{unpad(\rho(s)) : s \in L\}$. Moreover, an argument by induction on $\varphi \in P_\bullet$ establishes

Theorem 1. *Every $\varphi \in P_\bullet$ is stringwise P-represented by the regular language $\{unpad(\rho(s)) : s \in \hat{\mathcal{C}}(\varphi)_{\trianglerighteq}\}$.*

Remark The projection ρ drops $\top$. Writing $\mathsf{F}\varphi$ for $\top$ *until* φ as usual (and $\mathsf{P}\varphi$ for $\top$ *since* φ), one might try to replace $\boxed{\mathsf{F}q}\Box$ by $\boxed{q}\Box \mid \Box\boxed{\mathsf{F}q}$. But doing so in $\boxed{p, \mathsf{F}q}^+\boxed{r}\Box^*$ (where $L^+ \stackrel{\text{def}}{=} L^*L$) can lead to non-regularity, as intersection with the regular language $\boxed{p}^+\boxed{r}\boxed{q}^+$ yields the non-regular language

$$\{\boxed{p}^n\boxed{r}\boxed{q}^m : n \geq m \geq 1\}$$

with regular sublanguage $\boxed{p}^+\boxed{r}\boxed{q}$ obtained by $\trianglerighteq$-minimization and *unpad*.

4 Negation and paths for infinite strings

We turn next to negation and formulas such as $\mathsf{G}\varphi$ left out of Theorem 1.

4.1 Negation

The obvious constraints to associate with Boolean negation $\neg$

$$\langle V, x\rangle \models \neg\varphi \stackrel{\text{def}}{\iff} \text{not } \langle V, x\rangle \models \varphi$$

are non-contradiction $[\blacktriangleright]\,\overline{\boxed{\varphi, \neg\varphi}}$ and excluded middle

$$\Box \Rightarrow \boxed{\varphi} \mid \boxed{\neg\varphi}\ .$$

A popular alternative that we will adopt is to treat negation as a function $\varphi \mapsto \overline{\varphi}$ on formulas φ such that $\overline{\overline{\varphi}} = \varphi$ and $\overline{p} \in P$ for every $p \in P$ (if necessary, doubling P to $P \times \{+,-\}$ with $\overline{(p,+)} = (p,-)$ and $\overline{(p,-)} = (p,+)$). The functions (valuations) V are then required to satisfy $V(p) \cap V(\overline{p}) = \emptyset$, suggesting

$$[\blacktriangleright]\,\overline{\boxed{p, \overline{p}}}\ , \tag{10}$$

and $V(p) \cup V(\overline{p}) = \mathbb{Z}$. Every n-ary connective θ is paired with an n-ary connective $\overline{\theta}$ so that $\overline{\overline{\theta}} = \theta$ and

$$\overline{\theta(\varphi_1, \ldots, \varphi_n)} \stackrel{\text{def}}{=} \overline{\theta}(\overline{\varphi_1}, \ldots, \overline{\varphi_n})\ .$$

De Morgan's laws suggest $\overline{\vee} \stackrel{\text{def}}{=} \wedge$, $\overline{\theta} \stackrel{\text{def}}{=} \theta$ for $\theta \in \{\mathit{next},\ \mathit{prev}\}$, $\overline{\top} \stackrel{\text{def}}{=} \bot$ with

$$[\blacktriangleright]\,\overline{\boxed{\bot}} \tag{11}$$

(as $\langle V, x\rangle \not\models \bot$) and $\overline{\mathit{until}} \stackrel{\text{def}}{=} \mathit{release}$ where

$$\begin{aligned}\langle V, x\rangle \models \varphi\ \mathit{release}\ \psi \iff\ & (\forall y \geq x)\ \langle V, y\rangle \models \psi \text{ or} \\ & (\exists z < y)\ z \geq x \text{ and } \langle V, z\rangle \models \varphi\end{aligned}$$

covered by

$$\boxed{\varphi\ \mathit{release}\ \psi} \Rightarrow \boxed{\psi} \tag{12}$$

$$\boxed{\varphi\ \mathit{release}\ \psi}\Box \Rightarrow \boxed{\varphi}\Box \mid \Box\boxed{\varphi\ \mathit{release}\ \psi}\ . \tag{13}$$

We treat $\overline{\mathit{ntil}}$ and $\overline{\mathit{since}}$ similarly.

4.2 Paths

$\mathsf{G}\varphi$ is $\bot\ \mathit{release}\ \varphi$ and amounts to the infinite conjunction

$$\varphi \wedge \mathit{next}(\varphi) \wedge \mathit{next}(\mathit{next}(\varphi)) \wedge \cdots$$

which we shall analyze as follows. Given a language L, we say a language X is an *L-path* if $\emptyset \neq X \subseteq L$ and

(i) for all $s \in X$, there exists $s' \in X$ such that $s' \trianglerighteq \Box^+ s \Box^+$
(ii) for all $s, s' \in X$, there exists $s'' \in X$ such that $s'' \underline{\blacktriangleright} s$ and $s'' \underline{\blacktriangleright} s'$.

For example, for each $s \in L$, $\Box^* s \Box^*$ is a $\Box^* L \Box^*$-path (although a $\Box^* L \Box^*$-path need not be a subset of $\Box^* L \Box^+$ or $\Box^+ L \Box^*$). An L-path X is said to be *principal* if for some string s, $X \subseteq \Box^* s \Box^*$.

Definition. A set L of now_Φ-pointed strings *pathwise Φ-represents* φ if φ is equivalent to the disjunction over L-paths X of conjunctions $\bigwedge\{fmla(s) : s \in X\}$ in that

$$\langle V, x\rangle \models \varphi \iff (\exists L\text{-path } X)(\forall s \in X)\ \langle V, x\rangle \models fmla(s)$$

for all $V : P \to Pow(\mathbb{Z})$ and $x \in \mathbb{Z}$.

We can then prove an analog of Theorem 1 for pathwise (as opposed to stringwise) P-representations of formulas from a set P_∞ extending $P_\bullet$ with dual connectives $\bot, \mathit{release}, \overline{\mathit{ntil}}, \overline{\mathit{since}}$ and $\overline{\mathit{sinc}}$. $\mathcal{C}(\varphi)$ is revised to $\mathcal{D}(\varphi)$, bringing in the two forms (10) and (11) of non-contradiction,

$$\mathcal{D}(p) \stackrel{\text{def}}{=} [\blacktriangleright]\overline{\boxed{now}\Box^*\boxed{now} \mid \boxed{p, \overline{p}}} \qquad \text{for } p \in P$$

$$\mathcal{D}(\varphi) \stackrel{\text{def}}{=} [\blacktriangleright]\overline{\boxed{now}\Box^*\boxed{now} \mid \boxed{\bot}} \qquad \text{for } \varphi \in \{\top, \bot\}\ ,$$

treating $\wedge, \vee, \mathit{next}, \mathit{prev}, \mathit{until}, \mathit{since}, \mathit{ntil}$ and sinc as does $\mathcal{C}$

$$\mathcal{D}(\varphi \wedge \psi) \stackrel{\text{def}}{=} \mathcal{D}(\varphi) \cap \mathcal{D}(\psi) \cap (\boxed{\varphi \wedge \psi} \Rightarrow \boxed{\varphi, \psi})$$

$$\mathcal{D}(\varphi \vee \psi) \stackrel{\text{def}}{=} \mathcal{D}(\varphi) \cap \mathcal{D}(\psi) \cap (\boxed{\varphi \vee \psi} \Rightarrow \boxed{\varphi} \| \boxed{\psi})$$

$$\mathcal{D}(\mathit{next}(\varphi)) \stackrel{\text{def}}{=} \mathcal{D}(\varphi) \cap (\boxed{\mathit{next}(\varphi)} \stackrel{\mathsf{a}}{\Rightarrow} \boxed{\varphi})$$

etc, and building (12) and (13) into *release*

$$\begin{aligned}\mathcal{D}(\varphi\ \mathit{release}\ \psi) \stackrel{\text{def}}{=}\ & \mathcal{D}(\varphi) \cap \mathcal{D}(\psi) \cap \\ & (\boxed{\varphi\ \mathit{release}\ \psi} \Rightarrow \boxed{\psi})\ \cap \\ & (\boxed{\varphi\ \mathit{release}\ \psi}\Box \Rightarrow \boxed{\varphi}\Box \mid \Box\boxed{\varphi\ \mathit{release}\ \psi})\end{aligned}$$

and similarly for $\overline{\mathit{since}}$. Finally, we set

$$\hat{\mathcal{D}}(\varphi) \stackrel{\text{def}}{=} \mathcal{D}(\varphi) \cap \langle\blacktriangleright\rangle\boxed{now, \varphi}$$

and refrain from the unpadding in Theorem 1.

Theorem 2. *Every $\varphi \in P_\infty$ is pathwise P-represented by the regular language $\{\rho(s) : s \in \hat{\mathcal{D}}(\varphi)_{\trianglerighteq}\}$.*

5 Conclusion

The main results of the preceding account of temporal propositions as regular languages are Theorems 1 and 2 (from §§3.2 and 4.2). The theorems essentially implement well-known tableau constructions for Linear Temporal Logic

[3] through finite-state methods. As we shall see next, these methods extend to constructs in Hybrid Logic [10]. Beyond any application to a particular formal system, the methods feature notions (such as bounded entailment $\vdash_C$ defined in §2.2) that are part of a tool-kit for an approach to natural language semantics representing events as strings so that entailments can be read directly off the event representations. We close by outlining an approach based on these methods to changes over time against an inertial background [11].

5.1 Hybrid Logic

A basic notion in Hybrid Logic is that of a *nominal*, the collection of which we shall assume form a designated subset $P_0 \subseteq P$ of atomic propositions that a valuation V is required to map to singleton sets

$$(\forall n \in P_0)\ \ V(n) \text{ has cardinality } 1\ .$$

The uniqueness requirement on nominals n is built into the language

$$[\underline{\blacktriangleright}]\ \overline{\boxed{n}\Box^*\boxed{n}}$$

which we may assume is part of the constraints for $n \in P_0$. For arbitrary languages L and L' over Σ, let us write $L \overset{\blacktriangleright}{\rightarrow} L'$ for the set of strings that weakly subsume L' whenever they weakly subsume L

$$L \overset{\blacktriangleright}{\rightarrow} L' \overset{\text{def}}{=} \{s \in \Sigma^* \ :\ \text{if } s\underline{\blacktriangleright}L \text{ then } s\underline{\blacktriangleright}L'\}\ .$$

The special case of $L' = \emptyset$ reduces to $[\underline{\blacktriangleright}]\ \overline{L}$

$$[\underline{\blacktriangleright}]\ \overline{L} = L \overset{\blacktriangleright}{\rightarrow} \emptyset\ .$$

The operation $\overset{\blacktriangleright}{\rightarrow}$ preserves regularity, as

$$\begin{aligned} L \overset{\blacktriangleright}{\rightarrow} L' &= \overline{\langle\underline{\blacktriangleright}\rangle L \cap \overline{\langle\underline{\blacktriangleright}\rangle L'}} \\ &= [\{(s,s) : s\underline{\blacktriangleright}L\}]\langle\underline{\blacktriangleright}\rangle L'\ . \end{aligned}$$

Forming $L \overset{\blacktriangleright}{\rightarrow} L'$ with $L' \neq \emptyset$ pays off when analyzing a couple of constructs, @ and E, in Hybrid Logic. These constructs allow us to say of a temporal proposition φ that it holds at a nominal n

$$\langle V, x\rangle \models @_n\varphi \iff^{\text{def}} \langle V, n_V\rangle \models \varphi \quad \text{where } V(n) = \{n_V\} \tag{14}$$

or that it holds somewhere

$$\langle V, x\rangle \models \mathsf{E}\varphi \iff^{\text{def}} (\exists y)\ \langle V, y\rangle \models \varphi\ . \tag{15}$$

We can capture (14) as

$$\boxed{@_n\varphi} \overset{\blacktriangleright}{\rightarrow} \boxed{n, \varphi}$$

and (15) as

$$\boxed{\mathsf{E}\varphi} \xrightarrow{\blacktriangleright} \boxed{\varphi}\ .$$

To define negation via De Morgan duals, we set

$$\overline{@} = @$$
$$\overline{\mathsf{E}} = \mathsf{A}$$

and associate with $\mathsf{A}\varphi$ the constraints

$$\Box^+\boxed{\mathsf{A}\varphi} \Rightarrow \boxed{\varphi}\Box^+$$
$$\boxed{\mathsf{A}\varphi} \Rightarrow \boxed{\varphi}$$
$$\boxed{\mathsf{A}\varphi}\Box^+ \Rightarrow \Box^+\boxed{\varphi}$$

supporting a reading of $\mathsf{A}\varphi$ as "at all times (the past, the present and the future), φ."

Another construct from Hybrid Logic is the binder $\downarrow$ that we will assume combines a nominal $n \in P_0$ with a temporal formula φ in which '$\downarrow n$' does *not* occur. The resulting formula $\downarrow n.\varphi$ is then interpreted according to

$$\langle V, x\rangle \models \downarrow n.\varphi \stackrel{\text{def}}{\iff} \langle V_{x/n}, x\rangle \models \varphi$$

where $V_{x/n}$ is V except that it maps the nominal n to $\{x\}$. The corresponding constraint is

$$\boxed{\downarrow n.\varphi} \Rightarrow \boxed{n, \varphi}$$

(with the proviso that '$\downarrow n$' does not occur in φ).

5.2 Non-monotonic inertial reasoning

Finally, consider a temporal formula φ that, in the absence of a force against it, persists over time. A simple way of formulating this idea is to introduce a temporal formula $\mathsf{f}\overline{\varphi}$ intuitively saying that "a force is applied to make $\overline{\varphi}$ true (at the next step)" so that the constraint

$$\boxed{\varphi}\Box \Rightarrow \Box\boxed{\varphi} \mid \boxed{\mathsf{f}\overline{\varphi}}\Box \tag{16}$$

can be read as: φ persists (to the next step) unless a force is applied against it. Turning the force around to one $\mathsf{f}\varphi$ *for* (rather than against) φ, we obtain the backward form of persistence

$$\Box\boxed{\varphi} \Rightarrow \boxed{\varphi}\Box \mid \boxed{\mathsf{f}\varphi}\Box \tag{17}$$

making φ persist backward unless it was previously forced [9]. Together, (16) and (17) imply "no change without force." Distinguishing $\mathsf{f}\varphi$ from $\mathsf{f}\overline{\varphi}$ allows us to formulate the constraint

$$\boxed{\mathsf{f}\varphi}\Box \Rightarrow \Box\boxed{\varphi} \mid \boxed{\mathsf{f}\overline{\varphi}}\Box \tag{18}$$

saying that an unopposed force for φ brings φ about at the next step. The non-determinism expressed in the righthand sides of (16), (17) and (18) by choice $\mid$ opens the door to non-monotonicity as soon as we apply bias to choosing between the opposite sides of $\mid$. For instance, the assumption

(†) no force is applied unless it is explicitly mentioned

boosts the inference

$$\boxed{\varphi}\Box \vdash_{(16)} \Box\boxed{\varphi} \mid \boxed{\mathsf{f}\overline{\varphi}}\Box$$

to:

(‡) from $\boxed{\varphi}\Box$, infer $\Box\boxed{\varphi}$

(as no force is mentioned in $\boxed{\varphi}\Box$). The inference (‡) is soft inasmuch as the principle (†) licensing it is. (‡) is non-monotonic in that we lose the conclusion $\Box\boxed{\varphi}$ if we enrich the premise $\boxed{\varphi}\Box$ to $\boxed{\varphi, \mathsf{f}\overline{\varphi}}\Box$ (which subsumes $\boxed{\varphi}\Box$).

More precisely, recall that

$$L \vdash_C L' \iff C \cap \langle\underline{\blacktriangleright}\rangle L \ \underline{\blacktriangleright}\ L' \ . \tag{19}$$

If in (19) we were to refine the Peirce product

$$\langle\underline{\blacktriangleright}\rangle L = L_\Box \ \& \ \Sigma^*$$

(where $L_\Box$ is $\Box^* unpad(L)\Box^*$) by &-superposing $L_\Box$ not with Σ^* but with a sublanguage H such as

$$Pow(\Phi - \{\mathsf{f}\varphi, \mathsf{f}\overline{\varphi}, \ldots\})^*$$

then there would be more languages L' such that

$$C \cap (L_\Box \ \& \ H) \ \underline{\blacktriangleright}\ L' \tag{20}$$

than such that $L \vdash_C L'$. As far as computability is concerned, the important point about (20) is that it is as much an inclusion between regular languages as $L \vdash_C L'$ is. What (20) offers is a handle H on what to &-superpose with $L_\Box$ before intersecting it with the constraints C to see what is weakly subsumed. Under (20), there are two distinct ways to enrich $L_\Box$: by intersection with hard constraints C and by superposition with permissible hypotheses H. The non-monotonicity in (‡) above can be traced to a choice in (†) of H short of the full space Σ^* of possibilities entertained in $\vdash_C$. Bias is injected into the choice

$$\Box\boxed{\varphi} \mid \boxed{\mathsf{f}\overline{\varphi}}\Box$$

by including the left side $\Box\boxed{\varphi}$ in H, while excluding the right side $\boxed{\mathsf{f}\overline{\varphi}}\Box$ from H. Equally, we could pick an H' that throws *out* $\Box\boxed{\varphi}$ and lets in $\boxed{\mathsf{f}\overline{\varphi}}\Box$ to *explain* the failure of φ to persist in $\boxed{\varphi}\Box$.

References

1. Beesley, K.R., Karttunen, L.: Finite State Morphology. CSLI Publications, Stanford (2003)
2. Emerson, E.A.: Temporal and modal logic. In Leeuwen, J.v., ed.: Handbook of Theoretical Computer Science. Volume B: Formal Methods and Semantics. MIT Press (1992) 995–1072
3. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (1999)
4. Fernando, T.: A finite-state approach to events in natural language semantics. Journal of Logic and Computation **14**(1) (2004) 79–92
5. Bouajjani, A., Jonsson, B., Nilsson, M., Touili, T.: Regular model checking. In: Computer Aided Verification. Springer LNCS 1855 (2000) 403–418
6. Brink, C., Britz, K., Schmidt, R.: Peirce algebras. Formal Aspects of Computing **6**(3) (1994) 339–358
7. Harel, D.: Dynamic logic. In Gabbay, D., Guenthner, F., eds.: Handbook of Philosophical Logic. Volume 2. Reidel, Dordrecht (1984) 497–604
8. Karttunen, L.: www.stanford.edu/~laurik/fsmbook/examples/Yale Shooting.html (2005)
9. Fernando, T.: Finite-state temporal projection. In: Proc. 11th International Conference on Implementation and Application of Automata. Springer LNCS 4094 (2006) 230–241
10. Areces, C., ten Cate, B.: Hybrid logics. In Blackburn, P., Wolter, F., van Benthem, J., eds.: Handbook of Modal Logics. X (2005) (In Preparation).
11. McCarthy, J., Hayes, P.: Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, M., Michie, D., eds.: Machine Intelligence 4. Edinburgh University Press (1969) 463–502