SynCoP – Combining Syntactic Tagging with Chunking Using Weighted Finite State Transducers

Jörg Didakowski

Berlin-Brandenburgische Akademie der Wissenschaften
Jägerstr. 22/23, 10117 Berlin

Abstract. This paper describes the key aspects of the system SynCoP (Syntactic Constraint Parser) developed at the Berlin-Brandenburgische Akademie der Wissenschaften. The parser allows to combine syntactic tagging and chunking by means of constraint grammar using weighted finite state transducers (WFST). Chunks are interpreted as local dependency structures within syntactic tagging. The linguistic theories are formulated by criteria which are formalized by a semiring; these criteria allow structural preferences and gradual grammaticality. The parser is essentially a cascade of WFSTs. To find the most likely syntactic readings a best-path search is used.

1 Introduction

In several natural language processing tasks such as information extraction and machine translation and especially in corpus linguistics information about syntactic structures is needed. The main interest lies in detecting syntactic relations between words. This is generally done by building *dependency structures* of sentences.

This paper presents an approach to *dependency parsing* which combines *chunking* ([1]) and *syntactic tagging* by means of *constraint grammar* ([2]). Chunks are interpreted here as local dependency structures within syntactic tagging; this approach is related to [3]. The advantages of these two linguistic theories are linked: robustness is achieved by local structures and underspecified dependencies and disambiguation is done by a greedy strategy and by a pattern preference strategy.

Previous work which implements chunking (e.g. [4]) and syntactic tagging (e.g. [5]) with finite state machines leads to some restrictions and problems: chunking is implemented with the *left-to-right, longest-match replacement operator* [5] with which chunks are marked by brackets and are disambiguated by a left-to-right, longest-match strategy if there are various chunking possibilities. But the operator restricts the analysis to unambiguous input; in case of ambiguous input the longest-match is calculated for each input string independently. Hence, to achieve unambiguous chunking an unambiguous input has to be used.[1]

[1] Additionally, sometimes the left-to-right constraint causes unexpected analyses.

Furthermore, the operator leads to large finite state machines which require the limitation of patterns (see [6]). Syntactic tagging is implemented as *finite-state intersection grammar* by means of the *restriction operator* [7], which implements constraints as elimination rules. With these rules readings can be eliminated in an ambiguous input. But this implementation lacks the possibility of violating or preferring (weighting) constraints. Thus the main problem of implementing chunking and syntactic tagging is the way of doing disambiguation.

In the new approach proposed in this paper disambiguation is done by *linguistic criteria*. These criteria are formalized by a *semiring* over weights of a *weighted finite state transducer* (WFST). Scored dependency structures are generated over an input by a cascade of WFSTs (i.e. WFSTs are applied sequentially). Then the structures can be ordered on the basis of their weights by means of linguistic criteria to extract the most likely syntactic readings. This approach solves the problems mentioned above.

The paper is organized as follows: Section 2 gives basic definitions and notations. Section 3 is a brief reminder of the used linguistic theories. In sections 4, 5 and 6 the implementation of chunking, syntactic tagging and their combination with linguistic criteria is presented. Finally, an overview of the system *SynCoP* which implements the approach is given in section 7.

2 Definitions and Notations

In our approach syntactic analyses are generated and scored over an input by a WFST representing a constraint grammar such that syntactic readings can be judged by linguistic criteria. A weighted finite state transducer $T = (\Sigma, \Delta, Q, q_0, F, E, \lambda, \rho)$ over a semiring S is an 8-tuple such that Σ is the finite input alphabet, Δ is the finite output alphabet, Q is the finite set of states, $q_0 \in Q$ is the start state, $F \subseteq Q$ is the set of final states, $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times S \times Q$ is the set of transitions, λ is the initial weight and $\rho : F \mapsto S$ is the final weight function mapping final states to elements in S.

In this paper individual linguistic criteria are formalized by the notion of a semiring. Let $S \neq \emptyset$ be a set and $\oplus$ (called addition) and $\otimes$ (called multiplication) binary operations on S, then $(S, \oplus, \otimes, \bar{0}, \bar{1})$ is called a semiring if $(S, \oplus, \bar{0})$ is a commutative monoid, $(S, \otimes, \bar{1})$ is a monoid and $\otimes$ distributes over $\oplus$. Linguistic criteria are represented by this structure. To judge syntactic readings via addition an additive idempotent semiring has to be used to create a partial order over S. Thus a partial order is defined by $(a \leq_S b) \Leftrightarrow (a \oplus b = a)$. Here $a \leq_S b$ means that a is "better" than b in respect to linguistic criteria.

It will be necessary to judge analyses by more than one linguistic criterion; the criteria are ranked by preference. To model this we define the *composition* of additive idempotent semirings as follows: if a linguistic preference $(S_1, \oplus_1, \otimes_1, \bar{0}_1, \bar{1}_1) \succ (S_2, \oplus_2, \otimes_2, \bar{0}_2, \bar{1}_2) \succ ... \succ (S_n, \oplus_n, \otimes_n, \bar{0}_n, \bar{1}_n)$ is given and if for each semiring a partial order is defined by $\oplus$, then:[2]

[2] The definition is similarly to the *cross product* of semirings.

$$\begin{aligned}
&(S, \oplus, \otimes, \bar{0}, \bar{1}) = \\
&(S_1, \oplus_1, \otimes_1, \bar{0}_1, \bar{1}_1) \circ (S_2, \oplus_2, \otimes_2, \bar{0}_2, \bar{1}_2) \circ ... \circ (S_n, \oplus_n, \otimes_n, \bar{0}_n, \bar{1}_n) = \\
&(S_1 \times S_2 \times ... \times S_n, \oplus, \otimes, (\bar{0}_1, \bar{0}_2, ..., \bar{0}_n), (\bar{1}_1, \bar{1}_2, ..., \bar{1}_n))
\end{aligned} \tag{1}$$

The composition $\circ$ is the vectorization of the individual domains. The operation $\otimes$ of the semiring $(S, \oplus, \otimes, \bar{0}, \bar{1})$ is defined as a vectorization too, if $(a_1, a_2, ..., a_n) \in S$ and $(b_1, b_2, ..., b_n) \in S$ are given:

$$(a_1, a_2, ..., a_n) \otimes (b_1, b_2, ..., b_n) = (a_1 \otimes_1 b_1, a_2 \otimes_2 b_2, ..., a_n \otimes_n b_n) \tag{2}$$

And finally, the operation $\oplus$ which compares syntactic readings is defined as follows:

$$(a_1, a_2, ..., a_n) \oplus (b_1, b_2, ..., b_n) =$$
$$\begin{cases}
(a_1, a_2, ..., a_n) & \text{if } (a_1, a_2, ..., a_n) = (b_1, b_2, ..., b_n) \\
(a_1, a_2, ..., a_n) & \text{if } a_1 = b_1 \text{ and } a_2 = b_2 \text{ and ... and } a_{k-1} = b_{k-1} \\
& \text{and } a_k \oplus_k b_k = a_k \\
& \text{with } k \leq n \text{ and } a_k \neq b_k \\
(b_1, b_2, ..., b_n) & \text{if } a_1 = b_1 \text{ and } a_2 = b_2 \text{ and ... and } a_{k-1} = b_{k-1} \\
& \text{and } a_k \oplus_k b_k = b_k \\
& \text{with } k \leq n \text{ and } a_k \neq b_k
\end{cases} \tag{3}$$

With this composition it is possible to combine ranked linguistic criteria represented by several semirings to one semiring. The resulting semiring is now additive idempotent as well and a partial order can be defined by $\oplus$.

Syntactic analyses are ordered by linguistic criteria according to their degree of grammatical acceptance. This is done by a simple comparison: a reading is better than another or not. That allows structural preferences and gradual grammaticality. Extracting the most likely readings in a WFST T is a classical best-path problem. Weights along a path of T are combined by multiplication and create costs. If several paths are in T their weight equals the addition of weights of the different paths, that means the "best" cost (see [8]). The most likely syntactic analyses are simply represented by paths which cause these "best" costs.

A constraint grammar R can be applied by composition in linear time according to the size of an input acceptor S. Here, the composition can be computed in time $O(|R||S|)$ where $|R|$ and $|S|$ denote the number of states of R and S respectively (cf. [9]). The application of the constraint grammar results in a WFST which is acyclic. The best-path search can be calculated in $O(|Q| + |E|)$ in the acyclic case if Q is the set of states and E is the set of transitions (see [8]). Thus, the worst case time and space complexity of the application of a constraint grammar is $O(|S|)$ (cf. [6]).

In the following the ENGCG tagset and the regular expression notation of [5] (slightly extended) are used.[3]

[3] See appendix for regular expression notation details. Here the precedence is defined top down. The distinction between the language A and the identity relation which maps every string of A into itself is ignored.

3 Linguistic Background

In this section a short summary of syntactic tagging by means of constraint grammar, chunking and the combination of these two theories is given.

A constraint grammar [2] consists of a set of *constraints*, which can be seen as rules which are applied on linear patterns. The analysis starts from a large number of alternative analyses (syntactic and morphological) that are reduced by the application of constraints. *Syntactic tags* are used here to mark dependency relations where every tag represents a special dependency relation within a clause. In the following example an analysis of the sentence *Bill saw the little dog* is given:[4]

Bill	N NOM SG	@SUBJ	
saw	V PAST	@+FMAINV	
the	DET	@DN>	(4)
little	A	@AN>	
dog	N NOM SG	@OBJ	

The left column of the example above corresponds to the input sentence, the middle column to the morphological analyses and the right column shows the syntactic functions.

Chunks [1] are local constituent structures which are built by two principles: *chunk connectedness* and *chunk inclusiveness*. Chunk connectedness means that *functional elements* (e.g. a function word or an empty word) have to be grouped with their selected *thematic elements* (e.g. a content word or other chunks) forming a chunk. In addition to that, chunk inclusiveness claims that every word has to belong to a chunk with the exception of a distinguished subset of function words, which can't be grouped to their selected thematic elements because of intervening chunks. This function words are called *orphaned words*. The following example shows an analysis by chunks of the sentence used above:[5]

$$[\text{Ø}_{Det}\ \text{Bill}][\text{Ø}_{Comp}\ \text{saw}][\text{the little dog}][\text{in}[\text{the park}]] \tag{5}$$

In our approach, chunks are integrated in the syntactic tagging formalism. To infer a dependency structure from a constituent structure, a transformation rule can be applied: a head α governs his dependent β, iff α is the syntactic head of the constituent and β is its complement. To apply this transformation rule the syntactic head within a chunk has to be marked by a tag. Furthermore chunks by themselves have to be marked by a syntactic tag to be integrated into the syntax of the syntactic tagging. The following example shows this integration:[6]

[Ø_{Det} Bill@HEAD]	N NOM SG	@SUBJ	
[Ø_{Comp} saw@HEAD]	V PAST	@+FMAINV	(6)
[the little dog@HEAD]	N NOM SG	@OBJ	

[4] syntactic functions: @+FMAINV = finite main verb, @SUBJ = subject, @OBJ = object, @DN> = determiner, @AN> = premodifying adjective

[5] Ø_{Det} and Ø_{Comp} are empty functional elements.

[6] The heads in chunks are marked by @HEAD; the morphological analyses and syntactic functions refer to these heads.

Additionally, orphaned words can also be linked to their chunk by a special tag. In some languages, for example German, many orphaned words occur. One can question if chunk inclusiveness holds in all languages. With linking this problem can be overcome (see [10]).

4 Syntactic Tagging with Linguistic Criteria

Our interpretation of constraint grammar is related to [11]. Instead of implementing constraints as elimination rules constraints are regarded as score rules. But in addition we use scores to punish and to support patterns.

The *constraint optimization criterion* which represents the constraint grammar principle is formulated as follows: the patterns which receive the best scores by the constraints of a constraint grammar are in this sense the most grammatical. The criterion is formalized by the max semiring $(\mathbb{R} \cup \{-\infty\}, max, +, -\infty, 0)$; here the score is coded by numbers: positive numbers support patterns gradually and negative numbers punish patterns gradually.

To implement a constraint which supports patterns with respect to their context the *optional score operator* is defined as follows (it is sufficient to assign the weights optionally):

$$\begin{aligned}
\mathtt{A}(\Rightarrow_{\omega})\ \mathtt{B}\ \_\ \mathtt{C} &=_{def} \mathtt{[?*\ B\ A\ C}\ \langle\omega\rangle\mathtt{]}^*\mathtt{?}^* \\
\mathtt{A}(\Rightarrow_{\omega})\ \mathtt{B}\ \_ &=_{def} \mathtt{[?*\ B\ A}\ \langle\omega\rangle\mathtt{]}^*\mathtt{?}^* \\
\mathtt{A}(\Rightarrow_{\omega})\ \_\ \mathtt{C} &=_{def} \mathtt{[?*\ A\ C}\ \langle\omega\rangle\mathtt{]}^*\mathtt{?}^*
\end{aligned} \tag{7}$$

In contrast to the restriction operator [7], the optional score operator does not need the complementation of its operands. Thus, the resulting WFSTs stay small and transductions can be performed.[7]

The weights have to be assigned obligatorily (concerning the domain) to implement a constraint which punishes patterns with respect to their context. Otherwise the constraints have no effect. Thus, the *mandatory score operator* is defined as follows:[8]

$$\begin{aligned}
\mathtt{A}\Rightarrow_{\omega}\ \mathtt{B}\ \_\ \mathtt{C} &=_{def} \mathtt{[}\sim\$Dom(\mathtt{B\ A\ C})\ \mathtt{B\ A\ C}\ \langle\omega\rangle\mathtt{]}^*\ \sim\$Dom(\mathtt{B\ A\ C}) \\
\mathtt{A}\Rightarrow_{\omega}\ \mathtt{B}\ \_ &=_{def} \mathtt{[}\sim\$Dom(\mathtt{B\ A})\ \mathtt{B\ A}\ \langle\omega\rangle\mathtt{]}^*\ \sim\$Dom(\mathtt{B\ A}) \\
\mathtt{A}\Rightarrow_{\omega}\ \_\ \mathtt{C} &=_{def} \mathtt{[}\sim\$Dom(\mathtt{A\ C})\ \mathtt{A\ C}\ \langle\omega\rangle\mathtt{]}^*\ \sim\$Dom(\mathtt{A\ C})
\end{aligned} \tag{8}$$

With this operator transductions can be performed, too. But the operator is defined by complementation, hence the operator causes larger WFSTs.

[7] Complementation presupposes deterministic finite state acceptors and the determinization of the complement acceptors shows an exponential behaviour concerning the number of states.

[8] The function *Dom* returns the domain of the constraint patterns. So the regular language which does not contain the constraint patterns in respect of the domain can be built by complementation.

To assign positive or negative potentials to patterns we complete the optional score operator and the mandatory score operator:

$$\begin{array}{lll} \mathtt{A}(\Rightarrow_{\omega})\ \_ & =_{def} & \mathtt{[?^{*}\ A}\ \langle\omega\rangle\mathtt{]^{*}?^{*}} \\ \mathtt{A}\Rightarrow_{\omega}\ \_ & =_{def} & \mathtt{[\sim\$}Dom(\mathtt{A})\ \mathtt{A}\ \langle\omega\rangle\mathtt{]^{*}}\ \mathtt{\sim\$}Dom(\mathtt{A}) \end{array} \tag{9}$$

In the following an example of a constraint and its application are given. Let ... *a move* ... be an input fragment. We define the constraint `[@.N.]`$(\Rightarrow_{10})$`[@.DET.]` `_` to analyse the input fragment as follows:[9][10]

$$\begin{array}{lll} \mathtt{@\ a} & \mathtt{DET\ @DN>} & \\ \mathtt{@\ move} & \mathtt{[[N\ [@SUBJ|@OBJ|@I\text{-}OBJ](}\langle 10\rangle\mathtt{)]|} & \\ & \mathtt{[V\ \ @+FMAINV]]} & \end{array} \tag{10}$$

With the introduced weight ⟨10⟩ the reading in which *move* is a verb can be suppressed. A constraint grammar is constructed by combining the initial and the context constraints by composition. The big advantage of this approach is the possibility to violate constraints; so the application of a constraint grammar to an input is never empty.

5 Chunking with Linguistic Criteria

The grouping of chunks is formalized on the POS level by patterns which include the functional element optionally and its selected thematic element and the elements which can occur between them obligatorily. These patterns are marked in the input by brackets representing the syntactic projections. This is done by the *optional insertion operator* which is defined as follows if `A` denotes the chunk pattern and `P` and `S` the brackets:

$$\mathtt{A}(\rightarrow)\mathtt{P}...\mathtt{S}\ =_{def}\ \mathtt{[?^{*}[0.x.P]A[0.x.S]]^{*}?^{*}} \tag{11}$$

Building chunks with this implementation is not definite because of optional chunking, optional functional elements, ambiguous possible selections and especially ambiguous input. The following example *this cell structure* which includes the POS sequence `[DET|PRON]N[N|V]` shows the ambiguous chunking, if the chunker `[[(DET)N*]|[PRON]]`(→)`%[` ... `%]` is applied:[11]

$$\begin{array}{ccc} \text{this} & \text{cell} & \text{structure} \\ \text{[this]} & \text{[cell]} & \text{[structure]} \\ \text{[this]} & \text{[cell} & \text{structure]} \\ \text{[this} & \text{cell} & \text{structure]} \\ & \ldots & \end{array} \tag{12}$$

[9] The macro "`.`" is defined as ∼`$@` and the symbol "`@`" is used to mark word borders.

[10] For the sake of readability the morphological features are left out in the example.

[11] The lexicalized multi-word unit *cell structure* is treated as a non-lexicalized multi-word unit like *president Kennedy* or *city Berlin* which can also be seen as one thematic element.

The problem of ambiguity arises, because the greedy definition of chunks by chunk connectedness and chunk inclusiveness is not implemented.

Our approach concerning longest match is based on the work of [6] and [12]. There, a longest match constraint is implemented using the weights of a WFST. In our approach, the longest match is formalized by the tropical semiring $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$ and is implemented as follows: intra-chunk symbols receive a high negative number to prefer *exhaustive grouping* and brackets receive a low positive number to prefer *large groupings*. Following the example above, words within chunks get the weight ⟨-1⟩ and bracket pairs the weight ⟨0.1⟩; the syntactic readings are ordered by their longest match satisfaction:

$$\begin{array}{ll} \text{-2.9} & \text{[this cell structure]} \\ \text{-2.8} & \text{[this] [cell structure]} \\ \text{-2.7} & \text{[this] [cell] [structure]} \\ & \quad ... \\ 0 & \text{this cell structure} \end{array} \tag{13}$$

The analysis on the top represents the longest match (with the weight -2.9). It is possible to disambiguate chunking with the help of this greedy strategy.

Note that this approach is inconsistent in one case: Let $\mathtt{a}^*$ be the chunk pattern and let $\mathtt{a}_1\ \mathtt{a}_2\ ...\ \mathtt{a}_n$ be the input. The result of the chunking contains the weighted strings $\mathtt{[}\ ...\ \mathtt{|}\ \langle\omega_1\rangle\mathtt{\%[a}_1\mathtt{\%]\%[a}_2\mathtt{\%]}\ ...\ \mathtt{\%[a}_n\mathtt{\%]}\ \mathtt{|}\ \langle\omega_2\rangle\mathtt{\%[a}_1\mathtt{a}_2\ ...\ \mathtt{a}_{n-1}\mathtt{\%]a}_n\ \mathtt{|}\ ...\ \mathtt{]}$; there exists an n with $\omega_1 > \omega_2$ but there also exists another n with $\omega_1 \leq \omega_2$. It is possible that the result of the multiplication of the bracket costs contains the inverse of the cost assigned to a symbol within chunks. Hence in case of ambiguous input problems can arise.

To avoid this problem we formulate the disambiguation by chunk connectedness and chunk inclusiveness as criteria. The two criteria are ranked by preference: chunk inclusiveness $\succ$ chunk connectedness. The criteria say that words should belong to chunks primarily and words should form large chunks secondarily; that means functional elements should be grouped to their selected thematic elements. The chunk connectedness criterion and the chunk inclusiveness criterion are formalized separately by the tropical semiring $(\mathbb{R} \cup \{\infty\}, min, +, \infty, 0)$. Following the approach above the criteria are implemented by assigning negative numbers to symbols within chunks in respect of chunk inclusiveness on the one hand and assigning positive numbers to brackets in respect of chunk connectedness on the other hand. The optional insertion operator is now adjusted as follows:

$$\mathtt{A}(\rightarrow)\mathtt{P...S} =_{def} \mathtt{[?}^*\mathtt{[0.x.P][A.o.[?}\langle -1, 0\rangle\mathtt{]}^*\mathtt{][0.x.S]}\langle 0, 1\rangle\mathtt{]}^*\mathtt{?}^* \tag{14}$$

This implementation works on ambiguous input and disambiguates the input locally according to the linguistic criteria.[12] No complementation operation is used, hence transductions can be performed and the resulting WFSTs stay small

[12] With the given linguistic criteria the acceptance of the syntactic readings [*this*] [*cell structure*] and [*this cell*] [*structure*] equals. Such ambiguities should not be solved within this approach; the remaining ambiguities have no consequences. But in [10]

in size. Furthermore, the chunking can be affected by a later step because all chunking possibilities are enhanced.

To build a chunker with several planes of projection, several chunk types are built by the optional insertion operator and combined by composition. If several chunk types should compete within one level they are combined by union. Then the Kleene star is applied.

6 Combining Syntactic Tagging with Chunking

To integrate chunking within syntactic tagging, the syntactic heads in chunks are marked and the chunk itself is labeled with potential syntactic functions. This is achieved by the optional insertion operator. The following example shows this:

$$\texttt{[DET N[0.x.@HEAD]]}(\rightarrow)\texttt{\%[... \%][@SUBJ|@OBJ|@IOBJ]} \qquad (15)$$

Constraints can refer to chunks by their brackets and functions.

Via constraints it should be possible to restrict chunking to resolve *garden-path effects* which result from the greedy implementation. The problem is comparable to the *late closure* parsing principle [13] and is shown by the following example (the example is taken from [14]):

1. *[the emergency crews] really hate is [family violence]
   → garden-path effect
2. [the emergency] [crews] really hate is [family violence]
   → resolved

In order to do this, the constraint optimization criterion has to be ranked over the criteria concerning chunking. But the chunking should not be restricted by constraints in every case. Hence we distinguish between two kinds of constraint optimization criteria: strong and weak. The following ranking is used: strong constraint optimization $\succ$ chunk inclusiveness $\succ$ chunk connectedness $\succ$ weak constraint optimization. Now an analogical semiring can be built by composition.

Consequently, a constraint grammar contains both chunking and constraints. Here it is possible to assign costs concerning the weak constraint optimization criterion to chunk internal patterns. The advantage of combining chunking and constraints is obvious: robustness is reached by underspecified and local structures and the input is disambiguated by chunking and by constraints. Thus very robust and rich parsing is possible while the WFSTs of a constraint grammar stay small in size.

7 The System

SynCoP (Syntactic Constraint Parser) depends on the *Potsdam Finite State Library (FSM<2.0>)* [15], which makes it possible to change semirings via a

a criterion is presented which implements a *left-to-right preference strategy* which eliminates these ambiguities.

template. The system consists of a *grammar compiler* and a *grammar applier*. The grammar compiler takes an XML specification which contains definitions concerning constraints and chunking and builds a constraint grammar which consists of a cascade of WFSTs. This constraint grammar is used by the grammar applier to parse ambiguous input which is the result of the *TAGH morphology* [16].[13]

So far our goal is not to eliminate all morphological ambiguities but to extract dependency structures. Material which is not integrated into the dependency structures is not disambiguated. Our current hand-written grammar for German newspaper texts is compiled to a cascade of five WFSTs:

(1) morphology interface (48 states, 12415 transitions)
(2) chunking (70501 states, 246535 transitions)
(3) local dependency structures (6573 states, 356750 transitions)
(4) embedded clauses (1609 states, 188570 transitions)
(5) main clauses (2515 states, 346788 transitions)

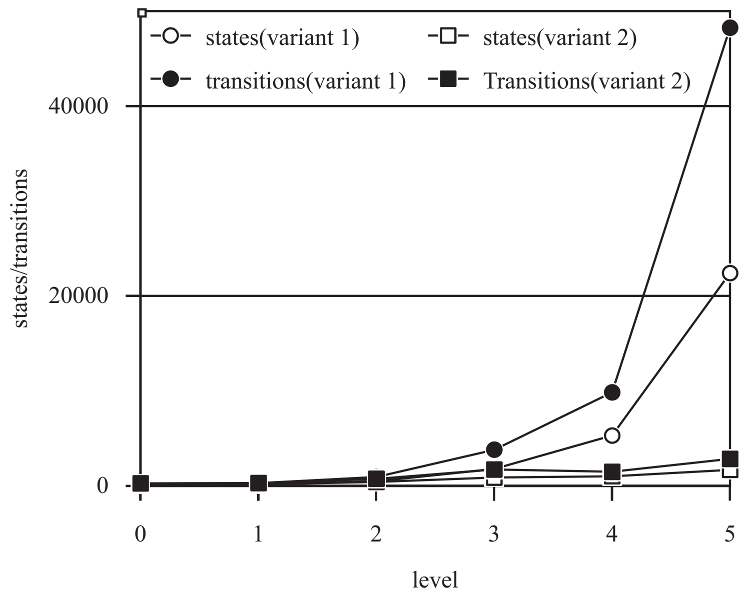

Fig. 1. Number of states and transitions in respect of the level of analysis

Level (1) does simple transductions mapping different POS tagsets and features and introduces possible main-clause and sub-clause borders and potential syntactic functions. Level (2) performs the chunking (NP-chunks, PP-chunks, AP-chunks) and chunk labeling with potential syntactic functions. Level (3) links

[13] The TAGH morphology contains information about lexicalized multi-word units. Hence, lexicalized and non-lexicalized multi-word units are discriminated.

orphaned words and disambiguates modifier functions or conjunctions which are related to chunks. The main reason for the large WFSTs concerning level (2) and (3) is the modeling of agreement in respect of case, number and gender. Finally level (4) disambiguates the head functions for embedded clauses and level (5) does the same for main clauses, while main-clauses and sub-clauses are detected.

During the analysis by our constraint grammar the resulting WFST grows level by level before the most likely syntactic readings are extracted by a best-path search. This is shown by the analysis of the German sentence *der Mann, der das Auto auf dem Parkplatz lieben wird, weint. (the man, which is going to love the car at the parking lot, cries.)*.[14] [15] The steps of this analysis concerning the number of states and transitions are shown in figure 1. Here, the number of states and transitions concerning the "normal" application of the constraint grammar is shown by variant 1. The input WFST has 158 states and 249 transitions. However the final result has 21406 states and 48135 transitions. The reason for that rapid growth is redundancy: ambiguous structures are looped through the levels instead of being resolved as early as possible.

To diminish this rapid growth we use a local disambiguation strategy. Material within chunks is never affected by constraints of higher levels; hence disambiguation can be performed within chunks without eliminating readings which are necessary for later steps. The same holds for sub-clauses. To do so, we implement a best-path search which refers to chunk and sub-clause brackets: after level (2) we disambiguate within chunks and after level (4) we disambiguate within sub-clauses. The amount of states and transitions concerning this strategy is shown in the figure above by variant 2. The resulting WFST has finally 1683 states and 2841 transitions; in level 4 the amount of transitions actually decreases.

8 Conclusion and Future Work

A new approach to robust dependency parsing which brings together syntactic tagging and chunking has been presented. Their combination is implemented by WFSTs over a semiring which represents several linguistic criteria. With these linguistic criteria disambiguation is done by a simple comparison concerning the degree of grammatical acceptance of syntactic analyses; this allows structural preferences and gradual grammaticality. It is possible to extend these linguistic criteria.

The System SynCoP is currently used for the construction of an engine analogical to the *Word Sketch Engine* [17].[16] Our word sketches show a promising

[14] PP-attachment is not covered by our constraint grammar yet.

[15] That sentence is analysed as follows: $[_{main_cl}[_{np}$der Mann@HEAD]@SUBJ, $[_{sub_cl}$der@SUBJ $[_{np}$das Auto@HEAD]@OBJ $[_{pp}$auf@HEAD $[_{np}$dem Parkplatz@HEAD]] lieben@-FMAINV wird@FAUXV], weint@+FMAINV .]

[16] A *word sketch* is a summary which is deduced from corpora showing grammatical and collocational behaviour of a word.

quality of annotation. An exhausting evaluation of our constraint grammar has not been done yet; this will be of interest in future work.

9 Appendix: Notations

(A)	option (union of A with epsilon)
$\sim$A	complement
$A	contains (all strings containing at least one A)
A^*	Kleene star
A^+	Kleene plus
A B	concatenation
A | B	union
A .x. B	crossproduct
A .o. B	composition
Dom(A)	the domain of a rational transduction
[and]	square brackets which group expressions
?	sigma
$?^*$	sigma star
0	epsilon
%	escape character
$\langle\omega\rangle$	weights

References

1. Abney, S.: Syntactic affixation and performance structures. In Bouchard, D., Leffel, K., eds.: Views on Phrase Structure. Kluwer (1990)
2. Karlsson, F.: Constraint grammar as a framework for parsing running text. In: Proceedings of the 13th International Conference on Computational Linguistics (COLING-90). Volume 3. (1990) 168–173
3. Aït-Mokhtar, S., Chanod, J.P.: Incremental finite state parsing. In: Proceedings of the 5th. International Conference on Applied Natural Language Processing (ANLP'97). (1997) 72–79
4. Grefenstette, G.: Light parsing as finite state filtering. In Kornai, A., ed.: Extended Finite-State Models of Language. Cambridge University Press (1999) 86–94
5. Karttunen, L.: Directed replacement. In Joshi, A., Palmer, M., eds.: Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics. (1996) 108–115
6. Hanneforth, T.: Longest-match pattern matching with weighted finite state automata. In: Proceedings of Finite-State Methods and Natural Language Processing (FSMNLP 05). (2005) 79–85
7. Koskenniemi, K.: Finite-state parsing and disambiguation. In: Proceedings of the the 13th International Conference on Computational Linguistics (COLING 90). Volume 2. (1990) 229–232
8. Mohri, M.: Semiring frameworks and algorithms for shortest-distance problems. Languages and Combinatorics **7**(3) (2002) 321–350
9. Tapanainen, P.: Applying a finite-state intersection grammar. In Roche, E., Schabes, Y., eds.: Finite-State language processing. MIT Press (1997) 311–327

10. Didakowski, J.: Robustes Parsing und Disambiguierung mit gewichteten Transduktoren. Volume 23. Linguistics in Potsdam (2005)
11. Tzoukerman, E., Radev, D.R.: Use of weighted finite state transducers in part of speech tagging. In Kornai, A., ed.: Extended Finite-State Models of Language. Cambridge University Press (1999) 193–207
12. Nasr, A., Volanschi, A.: Integrating a pos tagger and a chunker implemented as weighted finite state machines. In: Proceedings of Finite-State Methods and Natural Language Processing (FSMNLP 05). (2005) 167–178
13. Frazier, L., Clifton, Jr., C.: Construal. MIT Press (1996)
14. Abney, S.: Chunks and dependencies: Bringing processing evidence to bear on syntax. In Cole, J., Green, G., Morgan, J., eds.: Computational Linguistics and the Foundations of Linguistic Theory. CSLI (1995) 145–164
15. Hanneforth, T.: FSM<2.0> – C++ library for manipulating (weighted) finite automata. http://www.fsmlib.org (2005)
16. Geyken, A., Hanneforth, T.: Tagh: a complete morphology for german based on weighted finite state automaton. In: Proceedings of Finite-State Methods and Natural Language Processing (FSMNLP 05). (2005) 55–66
17. Kilgarriff, A., Rychly, P., Smrz, P., Tugwell, D.: The sketch engine. In: Proceedings of the Eleventh EURALEX International Congress. (2004) 105–116