Data-Driven Parsing using Probabilistic Linear Context-Free Rewriting Systems

Laura Kallmeyer*
Heinrich-Heine-Universität Düsseldorf

Wolfgang Maier**
Heinrich-Heine-Universität Düsseldorf

This paper presents the first efficient implementation of a weighted deductive CYK parser for Probabilistic Linear Context-Free Rewriting Systems (PLCFRSs). LCFRS, an extension of CFG, can describe discontinuities in a straightforward way and is therefore a natural candidate to be used for data-driven parsing. To speed up parsing, we use different context-summary estimates of parse items, some of them allowing for A parsing. We evaluate our parser with grammars extracted from the German NeGra treebank. Our experiments show that data-driven LCFRS parsing is feasible and yields output of competitive quality.*

1. Introduction

Recently, the challenges that a rich morphology poses for data-driven parsing have received growing interest. A direct effect of morphological richness is, for instance, data sparseness on a lexical level (Candito and Seddah 2010). A rather indirect effect is that morphological richness often relaxes word order constraints. The principal intuition is that a rich morphology encodes information that otherwise has to be conveyed by a particular word order. If, for instance, the case of a nominal complement is not provided by morphology, it has to be provided by the position of the complement relative to other complements in the sentence. Example (1) provides an example of case marking and free word order in German. In turn, in free word order languages, word order can encode information structure (Hoffman 1995).

(1) a. der kleine Junge$_{nom}$ schickt seiner Schwester$_{dat}$ den Brief$_{acc}$
the little boy sends his sister the letter

b. Other possible word orders:

(i) der kleine Junge$_{nom}$ schickt den Brief$_{acc}$ seiner Schwester$_{dat}$

(ii) seiner Schwester$_{dat}$ schickt der kleine Junge$_{nom}$ den Brief$_{acc}$

(iii) den Brief$_{acc}$ schickt der kleine Junge$_{nom}$ seiner Schwester$_{dat}$

* Institut für Sprache und Information, Universitätsstr. 1, D-40225 Düsseldorf, Germany.
E-mail: kallmeyer@phil.uni-duesseldorf.de.
** Institut für Sprache und Information, Universitätsstr. 1, D-40225 Düsseldorf, Germany.
E-mail: maierw@hhu.de.

Submission received: September 29, 2011; revised submission received: May 20, 2012; accepted for publication: August 3, 2012.

It is assumed that this relation between a rich morphology and free word order does not hold in both directions. Although it is generally the case that languages with a rich morphology exhibit a high degree of freedom in word order, languages with a free word order do not necessarily have a rich morphology. Two examples for languages with a very free word order are Turkish and Bulgarian. The former has a very rich and the latter a sparse morphology. See Müller (2002) for a survey of the linguistics literature on this discussion.

With a rather free word order, constituents and single parts of them can be displaced freely within the sentence. German, for instance, has a rich inflectional system and allows for a free word order, as we have already seen in Example (1): Arguments can be scrambled, and topicalizations and extrapositions underlie few restrictions. Consequently, discontinuous constituents occur frequently. This is challenging for syntactic description in general (Uszkoreit 1986; Becker, Joshi, and Rambow 1991; Bunt 1996; Müller 2004), and for treebank annotation in particular (Skut et al. 1997).

In this paper, we address the problem of data-driven parsing of discontinuous constituents on the basis of German. In this section, we inspect the type of data we have to deal with, and we describe the way such data are annotated in treebanks. We briefly discuss different parsing strategies for the data in question and motivate our own approach.

1.1 Discontinuous Constituents

Consider the sentences in Example (2) as examples for discontinuous constituents (taken from the German NeGra [Skut et al. 1997] and TIGER [Brants et al. 2002] treebanks). Example (2a) shows several instances of discontinuous VPs and Example (2b) shows a discontinuous NP. The relevant constituent is printed in italics.

(2) a. Fronting:

(i) *Darüber* muss *nachgedacht* werden. (NeGra)
Thereof must *thought* be
"One must think of that"

(ii) *Ohne internationalen Schaden* könne *sich* Bonn *von dem Denkmal* nicht
Without international damage could *itself* Bonn *from the monument* not
distanzieren, ... (TIGER)
distance
"Bonn could not distance itself from the monument without international damage."

(iii) *Auch* würden *durch die Regelung* nur *"ständig* neue Altfälle
Also would *through the regulation* only *"constantly* new old cases
entstehen". (TIGER)
emerge"
"Apart from that, the regulation would only constantly produce new old cases."

b. Extraposed relative clauses:

(i) ... ob auf deren Gelände der *Typ von Abstellanlage* gebaut
... whether on their terrain the *type of parking facility* built
werden könne, *der ...* (NeGra)
get could, *which ...*
"... whether one could build on their premises the type of parking facility, which ..."

Examples of other such languages are Bulgarian and Korean. Both show discontinuous constituents as well. Example (3a) is a Bulgarian example of a PP extracted out of an NP, taken from the BulTreebank (Osenova and Simov 2004), and Example (3b) is an example of fronting in Korean, taken from the Penn Korean Treebank (Han, Han, and Ko 2001).

(3) a. *Na kyshtata* toi popravi *pokriva*.
Of house-DET he repaired *roof*.
"It is the roof of the house *he repairs."*

b. *Gwon.han-ŭl* nu.ga *ka.ji.go* iss.ji?
Authority-OBJ who *has* not?
"Who has no authority?"

Discontinuous constituents are by no means limited to languages with freedom in word order. They also occur in languages with a rather fixed word order such as English, resulting from, for instance, long-distance movements. Examples (4a) and (4b) are examples from the Penn Treebank for long extractions resulting in discontinuous S categories and for discontinuous NPs arising from extraposed relative clauses, respectively (Marcus et al. 1994).

(4) a. Long Extraction in English:

(i) Those chains include Bloomingdale's, *which* Campeau recently said *it will sell*.

(ii) *What* should I *do.*

b. Extraposed nominal modifiers (relative clauses and PPs) in English:

(i) They sow *a row of male-fertile plants* nearby, *which then pollinate the male-sterile plants*.

(ii) *Prices* fell marginally *for fuel and electricity.*

1.2 Treebank Annotation and Data-Driven Parsing

Most constituency treebanks rely on an annotation backbone based on Context-Free Grammar (CFG). Discontinuities cannot be modeled with CFG, because they require a larger domain of locality than the one offered by CFG. Therefore, the annotation backbone based on CFG is generally augmented with a separate mechanism that accounts for the non-local dependencies. In the Penn Treebank (PTB), for example, trace nodes and co-indexation markers are used in order to establish additional implicit edges in the tree beyond the overt phrase structure. In TüBa-D/Z (Telljohann et al. 2012), a German Treebank, non-local dependencies are expressed via an annotation of topological fields (Höhle 1986) and special edge labels. In contrast, some other treebanks, among them NeGra and TIGER, give up the annotation backbone based on CFG and allow annotation with crossing branches (Skut et al. 1997). In such an annotation, non-local dependencies can be expressed directly by grouping all dependent elements under a single node. Note that both crossing branches and traces annotate long-distance dependencies in a linguistically meaningful way. A difference is, however, that crossing branches are less theory-dependent because they do not make any assumptions about the base positions of "moved" elements.

Examples for the different approaches of annotating discontinuities are given in Figures 1 and 2. Figure 1 shows the NeGra annotation of Example (2a-i) (left), and an

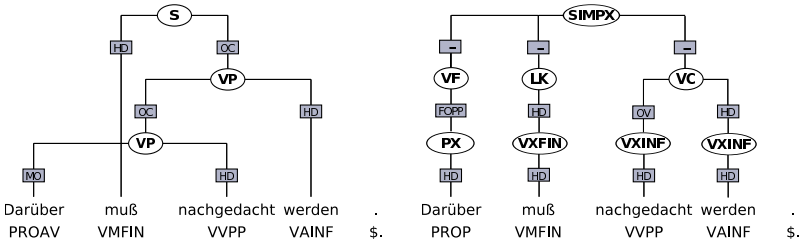


Figure 1
A discontinuous constituent. Original NeGra annotation (left) and a TüBa-D/Z-style annotation (right).

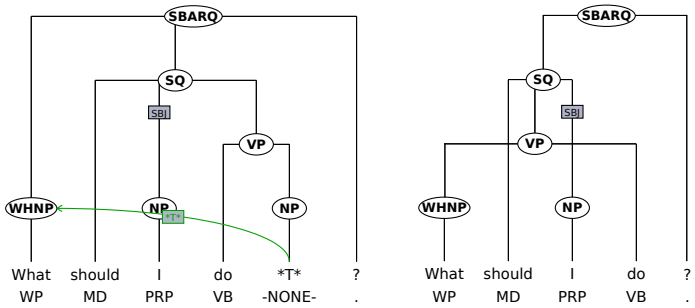


Figure 2
A discontinuous wh-movement. Original PTB annotation (left) and NeGra-style annotation (right).

annotation of the same sentence in the style of the TüBa-D/Z treebank (right). Figure 2 shows the PTB annotation of Example (4a-ii) (on the left, note that the directed edge from the trace to the WHNP element visualizes the co-indexation) together with a NeGra-style annotation of the same sentence (right).

In the past, data-driven parsing has largely been dominated by Probabilistic Context-Free Grammar (PCFG). In order to extract a PCFG from a treebank, the trees need to be interpretable as CFG derivations. Consequently, most work has excluded non-local dependencies; either (in PTB-like treebanks) by discarding labeling conventions such as the co-indexation of the trace nodes in the PTB, or (in NeGra/TIGER-like treebanks) by applying tree transformations, which resolve the crossing branches (e.g., Kübler 2005; Boyd 2007). Especially for the latter treebanks, such a transformation is problematic, because it generally is non-reversible and implies information loss.

Discontinuities are no minor phenomenon: Approximately 25% of all sentences in NeGra and TIGER have crossing branches (Maier and Lichte 2011). In the Penn Treebank, this holds for approximately 20% of all sentences (Evang and Kallmeyer 2011). This shows that it is important to properly treat such structures.

1.3 Extending the Domain of Locality

In the literature, different methods have been explored that allow for the use of non-local information in data-driven parsing. We distinguish two classes of approaches. The first class consists of approaches that aim at using formalisms which produce trees without crossing branches but provide a larger domain of locality than CFG—for instance, through complex labels (Hockenmaier 2003) or through the derivation

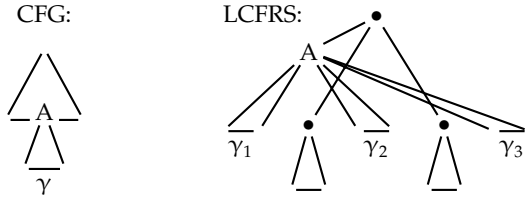


Figure 3
Different domains of locality.

mechanism (Chiang 2003). The second class, to which we contribute in this paper, consists of approaches that aim at producing trees which contain non-local information. Some methods realize the reconstruction of non-local information in a post- or pre-processing step to PCFG parsing (Johnson 2002; Dienes 2003; Levy and Manning 2004; Cai, Chiang, and Goldberg 2011). Other work uses formalisms that accommodate the direct encoding of non-local information (Plaehn 2004; Levy 2005). We pursue the latter approach.

Our work is motivated by the following recent developments. Linear Context-Free Rewriting Systems (LCFRSs) (Vijay-Shanker, Weir, and Joshi 1987) have been established as a candidate for modeling both discontinuous constituents and non-projective dependency trees as they occur in treebanks (Maier and Søgaard 2008; Kuhlmann and Satta 2009; Maier and Lichte 2011). LCFRSs are a natural extension of CFGs where the non-terminals can span tuples of possibly non-adjacent strings (see Figure 3). Because LCFRSs allow for binarization and CYK chart parsing in a way similar to CFGs, PCFG techniques, such as best-first parsing (Caraballo and Charniak 1998), weighted deductive parsing (Nederhof 2003), and A* parsing (Klein and Manning 2003a) can be transferred to LCFRS. Finally, as mentioned before, languages such as German have recently attracted the interest of the parsing community (Kübler and Penn 2008; Seddah, Kübler, and Tsarfaty 2010).

We bring together these developments by presenting a parser for Probabilistic LCFRS (PLCFRS), continuing the promising work of Levy (2005). Our parser produces trees with crossing branches and thereby accounts for syntactic long-distance dependencies while not making any additional assumptions concerning the position of hypothetical traces. We have implemented a CYK parser and we present several methods for context summary estimation of parse items. The estimates either act as figures-of-merit in a best-first parsing context or as estimates for A* parsing. A test on a real-world-sized data set shows that our parser achieves competitive results. To our knowledge, our parser is the first for the entire class of PLCFRS that has successfully been used for data-driven parsing.[1]

The paper is structured as follows. Section 2 introduces probabilistic LCFRS. Sections 3 and 4 present the binarization algorithm, the parser, and the outside estimates which we use to speed up parsing. In Section 5 we explain how to extract an LCFRS from a treebank and we present grammar refinement methods for these specific treebank grammars. Finally, Section 6 presents evaluation results and Section 7 compares our work to other approaches.

1 Parts of the results presented in this paper have been presented earlier. More precisely, in Kallmeyer and Maier (2010), we presented the general architecture of the parser and all outside estimates except the LN estimate from Section 4.4 which is presented in Maier, Kaeshammer, and Kallmeyer (2012). In Maier and Kallmeyer (2010) we have presented experiments with the relative clause split from Section 3.2. Finally, Maier (2010) contains the evaluation of the baseline (together with an evaluation using other metrics).

2. Probabilistic Linear Context-Free Rewriting Systems

2.1 Definition of PLCFRS

LCFRS (Vijay-Shanker, Weir, and Joshi 1987) is an extension of CFG in which a non-terminal can span not only a single string but a tuple of strings of size $k \geq 1$. k is thereby called its **fan-out**. We will notate LCFRS with the syntax of **Simple Range Concatenation Grammars** (SRCG) (Boullier 1998b), a formalism that is equivalent to LCFRS. A third formalism that is equivalent to LCFRS is **Multiple Context-Free Grammar** (MCFG) (Seki et al. 1991).

Definition 1 (LCFRS)
A **Linear Context-Free Rewriting System** (LCFRS) is a tuple $\langle N, T, V, P, S\rangle$ where

a) N is a finite set of non-terminals with a function dim: $N \to \mathbb{N}$ that determines the **fan-out** of each $A \in N$;

b) T and V are disjoint finite sets of terminals and variables;

c) $S \in N$ is the start symbol with $dim(S) = 1$;

d) P is a finite set of rules

$$A(\alpha_1, \ldots, \alpha_{dim(A)}) \to A_1(X_1^{(1)}, \ldots, X_{dim(A_1)}^{(1)}) \cdots A_m(X_1^{(m)}, \ldots, X_{dim(A_m)}^{(m)})$$

for $m \geq 0$ where $A, A_1, \ldots, A_m \in N$, $X_j^{(i)} \in V$ for $1 \leq i \leq m, 1 \leq j \leq dim(A_i)$ and $\alpha_i \in (T \cup V)^*$ for $1 \leq i \leq dim(A)$. For all $r \in P$, it holds that every variable X occurring in r occurs exactly once in the left-hand side and exactly once in the right-hand side of r.

A rewriting rule describes how the yield of the left-hand side non-terminal can be computed from the yields of the right-hand side non-terminals. The rules $A(ab, cd) \to \varepsilon$ and $A(aXb, cYd) \to A(X, Y)$ from Figure 4 for instance specify that (1) $\langle ab, cd\rangle$ is in the yield of A and (2) one can compute a new tuple in the yield of A from an already existing one by wrapping a and b around the first component and c and d around the second. A CFG rule $A \to BC$ would be written $A(XY) \to B(X)C(Y)$ as an LCFRS rule.

Definition 2 (Yield, language)
Let $G = \langle N, T, V, P, S\rangle$ be an LCFRS.

1. For every $A \in N$, we define the **yield** of A, $yield(A)$ as follows:
   a) For every rule $A(\vec{\alpha}) \to \varepsilon$, $\vec{\alpha} \in yield(A)$;

$$\begin{aligned} A(ab, cd) &\to \varepsilon \\ A(aXb, cYd) &\to A(X, Y) \\ S(XY) &\to A(X, Y) \end{aligned}$$

Figure 4
Sample LCFRS for $\{a^n b^n c^n d^n \mid n \geq 1\}$.

b) For every rule $A(\alpha_1, \ldots, \alpha_{dim(A)}) \rightarrow A_1(X_1^{(1)}, \ldots, X_{dim(A_1)}^{(1)}) \cdots A_m(X_1^{(m)}, \ldots, X_{dim(A_m)}^{(m)})$ and for all $\vec{\tau}_i \in yield(A_i)$ $(1 \leq i \leq m)$: $\langle f(\alpha_1), \ldots, f(\alpha_{dim(A)})\rangle \in yield(A)$ where f is defined as follows:

(i) $f(t) = t$ for all $t \in T$,

(ii) $f(X_j^{(i)}) = \vec{\tau}_i(j)$ for all $1 \leq i \leq m, 1 \leq j \leq dim(A_i)$ and

(iii) $f(xy) = f(x)f(y)$ for all $x, y \in (T \cup V)^+$.

We call f the **composition function** of the rule.

c) Nothing else is in $yield(A)$.

2. The language of G is then $L(G) = \{w \mid \langle w \rangle \in yield(S)\}$.

As an example, consider again the LCFRS in Figure 4. The last rule tells us that, given a pair in the yield of A, we can obtain an element in the yield of S by concatenating the two components. Consequently, the **language** generated by this grammar is $\{a^n b^n c^n d^n \mid n \geq 1\}$.

The terms of grammar fan-out and rank and the properties of monotonicity and ε-freeness will be referred to later and are therefore introduced in the following definition. They are taken from the LCFRS/MCFG terminology; the SRCG term for fan-out is **arity** and the property of being monotone is called **ordered** in the context of SRCG.

Definition 3
Let $G = \langle N, T, V, P, S\rangle$ be an LCFRS.

1. The **fan-out** of G is the maximal fan-out of all non-terminals in G.
2. Furthermore, the right-hand side length of a rewriting rule $r \in P$ is called the **rank** of r and the maximal rank of all rules in P is called the **rank** of G.
3. G is **monotone** if for every $r \in P$ and every right-hand side non-terminal A in r and each pair X_1, X_2 of arguments of A in the right-hand side of r, X_1 precedes X_2 in the right-hand side iff X_1 precedes X_2 in the left-hand side.
4. A rule $r \in P$ is called an ε-rule if one of the left-hand side components of r is ε.
   G is **ε-free** if it either contains no ε-rules or there is exactly one ε-rule $S(\varepsilon) \rightarrow \varepsilon$ and S does not appear in any of the right-hand sides of the rules in the grammar.

For every LCFRS there exists an equivalent LCFRS that is ε-free (Seki et al. 1991; Boullier 1998a) and monotone (Michaelis 2001; Kracht 2003; Kallmeyer 2010).

The definition of a probabilistic LCFRS is a straightforward extension of the definition of PCFG and thus it follows (Levy 2005; Kato, Seki, and Kasami 2006) that:

Definition 4 (PLCFRS)
A **probabilistic LCFRS** (PLCFRS) is a tuple $\langle N, T, V, P, S, p\rangle$ such that $\langle N, T, V, P, S\rangle$ is an LCFRS and $p : P \rightarrow [0..1]$ a function such that for all $A \in N$:

$$\Sigma_{A(\vec{x}) \rightarrow \vec{\Phi} \in P} p(A(\vec{x}) \rightarrow \vec{\Phi}) = 1$$

PLCFRS with non-terminals $\{S, A, B\}$, terminals $\{a\}$ and start symbol S:

$0.2 : S(X) \rightarrow A(X)$ $\quad 0.8 : S(XY) \rightarrow B(X, Y)$
$0.7 : A(aX) \rightarrow A(X)$ $\quad 0.3 : A(a) \rightarrow \varepsilon$
$0.8 : B(aX, aY) \rightarrow B(X, Y)$ $\quad 0.2 : B(a, a) \rightarrow \varepsilon$

Figure 5
Sample PLCFRS.

As an example, consider the PLCFRS in Figure 5. This grammar simply generates a^+. Words with an even number of *a*s and nested dependencies are more probable than words with a right-linear dependency structure. For instance, the word *aa* receives the two analyses in Figure 6. The analysis (a) displaying nested dependencies has probability 0.16 and (b) (right-linear dependencies) has probability 0.042.

3. Parsing PLCFRS

3.1 Binarization

Similarly to the transformation of a CFG into Chomsky normal form, an LCFRS can be binarized, resulting in an LCFRS of rank 2. As in the CFG case, in the transformation, we introduce a non-terminal for each right-hand side longer than 2 and split the rule into two rules, using this new intermediate non-terminal. This is repeated until all right-hand sides are of length 2. The transformation algorithm is inspired by Gómez-Rodríguez et al. (2009) and it is also specified in Kallmeyer (2010).

3.1.1 General Binarization. In order to give the algorithm for this transformation, we need the notion of a **reduction** of a vector $\vec{\alpha} \in [(T \cup V)^*]^i$ by a vector $\vec{x} \in V^j$ where all variables in $\vec{x}$ occur in $\vec{\alpha}$. A reduction is, roughly, obtained by keeping all variables in $\vec{\alpha}$ that are not in $\vec{x}$. This is defined as follows:

Definition 5 (Reduction)
Let $\langle N, T, V, P, S\rangle$ be an LCFRS, $\vec{\alpha} \in [(T \cup V)^*]^i$ and $\vec{x} \in V^j$ for some $i, j \in \mathbb{N}$.

Let $w = \vec{\alpha}_1\$ \ldots \$\vec{\alpha}_i$ be the string obtained from concatenating the components of $\vec{\alpha}$, separated by a new symbol $\$ \notin (V \cup T)$.

Let w' be the image of w under a homomorphism h defined as follows: $h(a) = \$$ for all $a \in T$, $h(X) = \$$ for all $X \in \{\vec{x}_1, \ldots \vec{x}_j\}$ and $h(y) = y$ in all other cases.

Let $y_1, \ldots y_m \in V^+$ such that $w' \in \$^* y_1 \$^+ y_2 \$^+ \ldots \$^+ y_m \$^*$. Then the vector $\langle y_1, \ldots y_m\rangle$ is the **reduction** of $\vec{\alpha}$ by $\vec{x}$.

For instance, $\langle aX_1, X_2, bX_3\rangle$ reduced with $\langle X_2\rangle$ yields $\langle X_1, X_3\rangle$ and $\langle aX_1X_2bX_3\rangle$ reduced with $\langle X_2\rangle$ yields $\langle X_1, X_3\rangle$ as well.

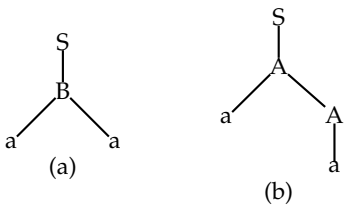


Figure 6
The two derivations of *aa*.

for all rules $r = A(\vec{\alpha}) \rightarrow A_0(\vec{\alpha_0}) \dots A_m(\vec{\alpha_m})$ in P with $m > 1$ **do**
  remove r from P
  $R := \emptyset$
  pick new non-terminals $C_1, \dots, C_{m-1}$
  add the rule $A(\vec{\alpha}) \rightarrow A_0(\vec{\alpha_0})C_1(\vec{\gamma_1})$ to R where $\vec{\gamma_1}$ is obtained by reducing $\vec{\alpha}$ with $\vec{\alpha_0}$
  for all i, $1 \leq i \leq m-2$ **do**
    add the rule $C_i(\vec{\gamma_i}) \rightarrow A_i(\vec{\alpha_i})C_{i+1}(\vec{\gamma_{i+1}})$ to R where $\vec{\gamma_{i+1}}$ is obtained by reducing $\vec{\gamma_i}$ with $\vec{\alpha_i}$
  end for
  add the rule $C_{m-1}(\vec{\gamma_{m-2}}) \rightarrow A_{m-1}(\vec{\alpha_{m-1}})A_m(\vec{\alpha_m})$ to R
  for every rule $r' \in R$ **do**
    replace right-hand side arguments of length > 1 with new variables (in both sides) and add the result to P
  end for
end for

Figure 7
Algorithm for binarizing an LCFRS.

The binarization algorithm is given in Figure 7. As already mentioned, it proceeds like the CFG binarization algorithm in the sense that for right-hand sides longer than 2, we introduce a new non-terminal that covers the right-hand side without the first element. Figure 8 shows an example. In this example, there is only one rule with a right-hand side longer than 2. In a first step, we introduce the new non-terminals and rules that binarize the right-hand side. This leads to the set R. In a second step, before adding the rules from R to the grammar, whenever a right-hand side argument contains several variables, these are collapsed into a single new variable.

The equivalence of the original LCFRS and the binarized grammar is rather straightforward. Note, however, that the fan-out of the LCFRS can increase.

The binarization depicted in Figure 7 is deterministic in the sense that for every rule that needs to be binarized, we choose unique new non-terminals. Later, in Section 5.3.1, we will introduce additional factorization into the grammar rules that reduces the set of new non-terminals.

3.1.2 Minimizing Fan-Out and Number of Variables. In LCFRS, in contrast to CFG, the order of the right-hand side elements of a rule does not matter for the result of a derivation.

Original LCFRS:
$S(XYZUVW) \rightarrow A(X,U)B(Y,V)C(Z,W)$
$A(aX,aY) \rightarrow A(X,Y) \quad A(a,a) \rightarrow \varepsilon$
$B(bX,bY) \rightarrow B(X,Y) \quad B(b,b) \rightarrow \varepsilon$
$C(cX,cY) \rightarrow C(X,Y) \quad C(c,c) \rightarrow \varepsilon$

Rule with right-hand side of length > 2: $S(XYZUVW) \rightarrow A(X,U)B(Y,V)C(Z,W)$
For this rule, we obtain
$R = \{S(XYZUVW) \rightarrow A(X,U)C_1(YZ,VW), C_1(YZ,VW) \rightarrow B(Y,V)C(Z,W)\}$

Equivalent binarized LCFRS:
$S(XPUQ) \rightarrow A(X,U)C_1(P,Q)$
$C_1(YZ,VW) \rightarrow B(Y,V)C(Z,W)$
$A(aX,aY) \rightarrow A(X,Y) \quad A(a,a) \rightarrow \varepsilon$
$B(bX,bY) \rightarrow B(X,Y) \quad B(b,b) \rightarrow \varepsilon$
$C(cX,cY) \rightarrow C(X,Y) \quad C(c,c) \rightarrow \varepsilon$

Figure 8
Sample binarization of an LCFRS.

Therefore, we can reorder the right-hand side of a rule before binarizing it. In the following, we present a binarization order that yields a minimal fan-out and a minimal variable number per production and binarization step. The algorithm is inspired by Gómez-Rodríguez et al. (2009) and has first been published in this version in Kallmeyer (2010). We assume that we are only considering partitions of right-hand sides where one of the sets contains only a single non-terminal.

For a given rule $c = A_0(\vec{x_0}) \rightarrow A_1(\vec{x_1}) \dots A_k(\vec{x_k})$, we define the **characteristic string** $s(c, A_i)$ of the A_i-reduction of c as follows: Concatenate the elements of $\vec{x_0}$, separated with new additional symbols \$ while replacing every component from $\vec{x_i}$ with a \$. We then define the arity of the characteristic string, $dim(s(c, A_i))$, as the number of maximal substrings $x \in V^+$ in $s(A_i)$. Take, for example, a rule $c = VP(X, YZU) \rightarrow VP(X, Z)V(Y)N(U)$. Then $s(c, VP) =$\$\$$Y$\$$U$, $s(c, V) = X$\$\$$ZU$.

Figure 9 shows how in a first step, for a given rule r with right-hand side length > 2, we determine the optimal candidate for binarization based on the characteristic string $s(r, B)$ of some right-hand side non-terminal B and on the fan-out of B: On all right-hand side predicates B we check for the maximal fan-out (given by $dim(s(r, B))$) and the number of variables ($dim(s(r, B)) + dim(B)$) we would obtain when binarizing with this predicate. This check provides the optimal candidate. In a second step we then perform the same binarization as before, except that we use the optimal candidate now instead of the first element of the right-hand side.

3.2 The Parser

We can assume without loss of generality that our grammars are ε-free and monotone (the treebank grammars with which we are concerned all have these properties) and that they contain only binary and unary rules. Furthermore, we assume POS tagging to be done before parsing. POS tags are non-terminals of fan-out 1. Finally, according to our grammar extraction algorithm (see Section 5.1), a separation between two components always means that there is actually a non-empty gap in between them. Consequently, two different components in a right-hand side can never be adjacent in the same component of the left-hand side. The rules are then either of the form $A(a) \rightarrow \varepsilon$ with A a POS tag and $a \in T$ or of the form $A(\vec{x}) \rightarrow B(\vec{x})$ or $A(\vec{\alpha}) \rightarrow B(\vec{x})C(\vec{y})$ where $\vec{\alpha} \in (V^+)^{dim(A)}$, $\vec{x} \in V^{dim(B)}$, $\vec{y} \in V^{dim(C)}$, that is, only the rules for POS tags contain terminals in their left-hand sides.

cand = 0
fan-out = number of variables in r
vars = number of variables in r
for all $i = 0$ **to** m **do**
  cand-fan-out = $dim(s(r, A_i))$;
  if *cand-fan-out* < *fan-out* and $dim(A_i)$ < *fan-out* **then**
    fan-out = $max(\{$*cand-fan-out*, $dim(A_i)\})$;
    vars = *cand-fan-out* + $dim(A_i)$;
    cand = i;
  else if *cand-fan-out* ≤ *fan-out*, $dim(A_i) \leq$ *fan-out* and *cand-fan-out* + $dim(A_i)$ < *vars* **then**
    fan-out = $max(\{$*cand-fan-out*, $dim(A_i)\})$;
    vars = *cand-fan-out* + $dim(A_i)$;
    cand = i
  end if
end for

Figure 9
Optimized version of the binarization algorithm, determining binarization order.

During parsing we have to link the terminals and variables in our LCFRS rules to portions of the input string. For this purpose we need the notions of ranges, range vectors, and rule instantiations. A range is a pair of indices that characterizes the span of a component within the input. A range vector characterizes a tuple in the yield of a non-terminal. A rule instantiation specifies the computation of an element from the left-hand side yield from elements in the yields of the right-hand side non-terminals based on the corresponding range vectors.

Definition 6 (Range)
Let $w \in T^*$ with $w = w_1 \dots w_n$ where $w_i \in T$ for $1 \leq i \leq n$.

1. $Pos(w) := \{0, \dots, n\}$.
2. We call a pair $\langle l, r \rangle \in Pos(w) \times Pos(w)$ with $l \leq r$ a **range** in w. Its **yield** $\langle l, r \rangle(w)$ is the substring $w_{l+1} \dots w_r$.
3. For two ranges $\rho_1 = \langle l_1, r_1 \rangle, \rho_2 = \langle l_2, r_2 \rangle$, if $r_1 = l_2$, then the concatenation of ρ_1 and ρ_2 is $\rho_1 \cdot \rho_2 = \langle l_1, r_2 \rangle$; otherwise $\rho_1 \cdot \rho_2$ is undefined.
4. A $\vec{\rho} \in (Pos(w) \times Pos(w))^k$ is a k-dimensional **range vector** for w iff $\vec{\rho} = \langle \langle l_1, r_1 \rangle, \dots, \langle l_k, r_k \rangle \rangle$ where $\langle l_i, r_i \rangle$ is a range in w for $1 \leq i \leq k$.

We now define instantiations of rules with respect to a given input string. This definition follows the definition of **clause instantiations** from Boullier (2000). An instantiated rule is a rule in which variables are consistently replaced by ranges. Because we need this definition only for parsing our specific grammars, we restrict ourselves to ε-free rules containing only variables.

Definition 7 (Rule instantiation)
Let $G = (N, T, V, P, S)$ be an ε-free monotone LCFRS. For a given rule $r = A(\vec{\alpha}) \rightarrow A_1(\vec{x_1}) \cdots A_m(\vec{x_m}) \in P$ $(0 < m)$ that does not contain any terminals,

1. an **instantiation** with respect to a string $w = t_1 \dots t_n$ consists of a function $f : V \rightarrow \{\langle i, j \rangle \mid 1 \leq i \leq j \leq |w|\}$ such that for all x, y adjacent in one of the elements of $\vec{\alpha}$, $f(x) \cdot f(y)$ must be defined; we then define $f(xy) = f(x) \cdot f(y)$,
2. if f is an instantiation of r, then $A(f(\vec{\alpha})) \rightarrow A_1(f(\vec{x_1})) \cdots A_m(f(\vec{x_m}))$ is an **instantiated rule** where $f(\langle x_1, \dots, x_k \rangle) = \langle f(x_1), \dots, f(x_k) \rangle$.

We use a probabilistic version of the CYK parser from Seki et al. (1991). The algorithm is formulated using the framework of parsing as deduction (Pereira and Warren 1983; Shieber, Schabes, and Pereira 1995; Sikkel 1997), extended with weights (Nederhof 2003). In this framework, a set of weighted items representing partial parsing results is characterized via a set of deduction rules, and certain items (the goal items) represent successful parses.

During parsing, we have to match components in the rules we use with portions of the input string. For a given input w, our items have the form $[A, \vec{\rho}]$ where $A \in N$ and $\vec{\rho}$ is a range vector that characterizes the span of A. Each item has a weight *in* that encodes the Viterbi inside score of its best parse tree. More precisely, we use the log probability $\log(p)$ where p is the probability.

The first rule (**scan**) tells us that the POS tags that we receive as inputs are given. Consequently, they are axioms; their probability is 1 and their weight therefore 0. The

Scan: $\dfrac{}{0 : [A, \langle\langle i, i+1\rangle\rangle]}$ A is the POS tag of w_{i+1}

Unary: $\dfrac{in : [B, \vec{\rho}]}{in + log(p) : [A, \vec{\rho}]}$ $p : A(\vec{\alpha}) \rightarrow B(\vec{\alpha}) \in P$

Binary: $\dfrac{in_B : [B, \vec{\rho_B}], in_C : [C, \vec{\rho_C}]}{in_B + in_C + log(p) : [A, \vec{\rho_A}]}$ $p : A(\vec{\rho_A}) \rightarrow B(\vec{\rho_B})C(\vec{\rho_C})$ is an instantiated rule

Goal: $[S, \langle\langle 0, n\rangle\rangle]$

Figure 10
Weighted CYK deduction system.

second rule, **unary**, is applied whenever we have found the right-hand side of an instantiation of a unary rule. In our grammar, terminals only occur in rules with POS tags and the grammar is ordered and ε-free. Therefore, the components of the yield of the right-hand side non-terminal and of the left-hand side terminals are the same. The rule **binary** applies an instantiated rule of rank 2. If we already have the two elements of the right-hand side, we can infer the left-hand side element. In both cases, **unary** and **binary**, the probability p of the new rule is multiplied with the probabilities of the antecedent items (which amounts to summing up the antecedent weights and $log(p)$).

We perform weighted deductive parsing, based on the deduction system from Figure 10. We use a chart $\mathcal{C}$ and an agenda $\mathcal{A}$, both initially empty, and we proceed as in Figure 11. Because for all our deduction rules, the weight functions f that compute the weight of a consequent item from the weights of the antecedent items are monotone non-increasing in each variable, the algorithm will always find the best parse without the need of exhaustive parsing. All new items that we deduce involve at least one of the agenda items as an antecedent item. Therefore, whenever an item is the best in the agenda, we can be sure that we will never find an item with a better (i.e., higher) weight. Consequently, we can safely store this item in the chart and, if it is a goal item, we have found the best parse.

As an example consider the development of the agenda and the chart in Figure 12 when parsing *aa* with the PLCFRS from Figure 5, transformed into a PLCFRS with pre-terminals and binarization (i.e., with a POS tag T_a and a new binarization non-terminal B'). The new PLCFRS is given in Figure 13.

In this example, we find a first analysis for the input (a goal item) when combining an A with span $\langle\langle 0, 2\rangle\rangle$ into an S. This S has however a rather low probability and is therefore not on top of the agenda. Later, when finding the better analysis, the weight

add SCAN results to $\mathcal{A}$
while $\mathcal{A} \neq \emptyset$
  remove best item $x : I$ from $\mathcal{A}$
  add $x : I$ to $\mathcal{C}$
  if I goal item
  then stop and output true
  else
    for all $y : I'$ deduced from $x : I$ and items in $\mathcal{C}$:
      if there is no z with $z : I' \in \mathcal{C} \cup \mathcal{A}$
      then add $y : I'$ to $\mathcal{A}$
      else if $z : I' \in \mathcal{A}$ for some z
        then update weight of I' in $\mathcal{A}$ to $max(y, z)$

Figure 11
Weighted deductive parsing.

chart	agenda
	$0 : [T_a, \langle 0,1\rangle], 0 : [T_a, \langle 1,2\rangle]$
$0 : [T_a, \langle 0,1\rangle]$	$0 : [T_a, \langle 1,2\rangle], -0.5 : [A, \langle 0,1\rangle]$
$0 : [T_a, \langle 0,1\rangle], 0 : [T_a, \langle 1,2\rangle]$	$-0.5 : [A, \langle 0,1\rangle], -0.5 : [A, \langle 1,2\rangle],$ $-0.7 : [B, \langle 0,1\rangle, \langle 1,2\rangle]$
$0 : [T_a, \langle 0,1\rangle], 0 : [T_a, \langle 1,2\rangle],$ $-0.5 : [A, \langle 0,1\rangle]$	$-0.5 : [A, \langle 1,2\rangle], -0.7 : [B, \langle 0,1\rangle, \langle 1,2\rangle],$ $-1.2 : [S, \langle 0,1\rangle]$
$0 : [T_a, \langle 0,1\rangle], 0 : [T_a, \langle 1,2\rangle],$ $0.5 : [A, \langle 0,1\rangle], -0.5 : [A, \langle 1,2\rangle]$	$-0.65 : [A, \langle 0,2\rangle], -0.7 : [B, \langle 0,1\rangle, \langle 1,2\rangle],$ $-1.2 : [S, \langle 0,1\rangle], -1.2 : [S, \langle 1,2\rangle]$
$0 : [T_a, \langle 0,1\rangle], 0 : [T_a, \langle 1,2\rangle],$ $-0.5 : [A, \langle 0,1\rangle], -0.5 : [A, \langle 1,2\rangle],$ $-0.65 : [A, \langle 0,2\rangle]$	$-0.7 : [B, \langle 0,1\rangle, \langle 1,2\rangle], -1.2 : [S, \langle 0,1\rangle],$ $-1.2 : [S, \langle 1,2\rangle], -1.35 : [S, \langle 0,2\rangle]$
$0 : [T_a, \langle 0,1\rangle], 0 : [T_a, \langle 1,2\rangle],$ $-0.5 : [A, \langle 0,1\rangle], -0.5 : [A, \langle 1,2\rangle],$ $-0.65 : [A, \langle 0,2\rangle], -0.7 : [B, \langle 0,1\rangle, \langle 1,2\rangle]$	$-0.8 : [S, \langle 0,2\rangle], -1.2 : [S, \langle 0,1\rangle],$ $-1.2 : [S, \langle 1,2\rangle]$

Figure 12
Parsing of *aa* with the grammar from Figure 5.

PLCFRS with non-terminals $\{S, A, B, B', T_a\}$, terminals $\{a\}$ and start symbol S:

$0.2 : S(X) \to A(X)$	$0.8 : S(XY) \to B(X, Y)$
$0.7 : A(XY) \to T_a(X)A(Y)$	$0.3 : A(X) \to T_a(X)$
$0.8 : B(ZX, Y) \to T_a(Z)B'(X, Y)$	$1 : B'(X, UY) \to B(X, Y)T_a(U)$
$0.2 : B(X, Y) \to T_a(X)T_a(Y)$	$1 : T_a(a) \to \varepsilon$

Figure 13
Sample binarized PLCFRS (with pre-terminal T_a).

of the S item in the agenda is updated and then the goal item is the top agenda item and therefore parsing has been successful.

Note that, so far, we have only presented the recognizer. In order to extend it to a parser, we do the following: Whenever we generate a new item, we store it not only with its weight but also with backpointers to its antecedent items. Furthermore, whenever we update the weight of an item in the agenda, we also update the backpointers. In order to read off the best parse tree, we have to start from the goal item and follow the backpointers.

4. Outside Estimates

So far, the weights we use give us only the Viterbi inside score of an item. In order to speed up parsing, we add the estimate of the costs for completing the item into a goal item to its weight—that is, to the weight of each item in the agenda, we add an estimate of its Viterbi outside score[2] (i.e., the logarithm of the estimate). We use **context summary estimates**. A context summary is an equivalence class of items for which we can compute the actual outside scores. Those scores are then used as estimates. The challenge is to choose the estimate general enough to be efficiently computable and specific enough to be helpful for discriminating items in the agenda.

2 Note that just as Klein and Manning (2003a), we use the terms **inside score** and **outside score** to denote the Viterbi inside and outside scores. They are not to be confused with the actual inside or outside probability.

Admissibility and **monotonicity** are two important conditions on estimates. All our outside estimates are **admissible** (Klein and Manning 2003a), which means that they never underestimate the actual outside score of an item. In other words, they are too optimistic about the costs of completing the item into an S item spanning the entire input. For the full SX estimate described in Section 4.1 and the SX estimate with span and sentence length in Section 4.4, the monotonicity is guaranteed and we can do true A* parsing as described by Klein and Manning. Monotonicity means that for each antecedent item of a rule it holds that its weight is greater than or equal to the weight of the consequent item. The estimates from Sections 4.2 and 4.3 are not monotonic. This means that it can happen that we deduce an item I_2 from an item I_1 where the weight of I_2 is greater than the weight of I_1. The parser can therefore end up in a local maximum that is not the global maximum we are searching for. In other words, those estimates are only **figures of merit** (FOM).

All outside estimates are computed off-line for a certain maximal sentence length len_{max}.

4.1 Full SX Estimate

The full SX estimate is a PLCFRS adaption of the **SX estimate** of Klein and Manning (2003a) (hence the name). For a given sentence length n, the estimate gives the maximal probability of completing a category X with a span ρ into an S with span $\langle\langle 0, n\rangle\rangle$.

For its computation, we need an estimate of the inside score of a category C with a span ρ, regardless of the actual terminals in our input. This inside estimate is computed as shown in Figure 14. Here, we do not need to consider the number of terminals outside the span of C (to the left or right or in the gaps), because they are not relevant for the inside score. Therefore the items have the form $[A, \langle l_1, \ldots, l_{dim(A)}\rangle]$, where A is a non-terminal and l_i gives the length of its ith component. It holds that

$$\Sigma_{1 \leq i \leq dim(A)} l_i \leq len_{max} - dim(A) + 1$$

because our grammar extraction algorithm ensures that the different components in the yield of a non-terminal are never adjacent. There is always at least one terminal in between two different components that does not belong to the yield of the non-terminal.

The first rule in Figure 14 tells us that POS tags always have a single component of length 1; therefore this case has probability 1 (weight 0). The rules **unary** and **binary** are roughly like the ones in the CYK parser, except that they combine items with length information. The rule **unary** for instance tells us that if the log of the probability of building $[B, \vec{l}]$ is greater or equal to in and if there is a rule that allows to deduce an

POS tags: $\dfrac{}{0 : [A, \langle 1\rangle]}$ A a POS tag

Unary: $\dfrac{in : [B, \vec{l}]}{in + log(p) : [A, \vec{l}]}$ $p : A(\vec{\alpha}) \rightarrow B(\vec{\alpha}) \in P$

Binary: $\dfrac{in_B : [B, \vec{l_B}], in_C : [C, \vec{l_C}]}{in_B + in_C + log(p) : [A, \vec{l_A}]}$

where $p : A(\vec{\alpha_A}) \rightarrow B(\vec{\alpha_B})C(\vec{\alpha_C}) \in P$ and the following holds: we define $\mathcal{B}(i)$ as $\{1 \leq j \leq dim(B) \mid \vec{\alpha_B}(j)$ occurs in $\vec{\alpha_A}(i)\}$ and $\mathcal{C}(i)$ as $\{1 \leq j \leq dim(C) \mid \vec{\alpha_C}(j)$ occurs in $\vec{\alpha_A}(i)\}$.
Then for all i, $1 \leq i \leq dim(A)$: $\vec{l_A}(i) = \Sigma_{j \in \mathcal{B}(i)} \vec{l_B}(j) + \Sigma_{j \in \mathcal{C}(i)} \vec{l_C}(j)$.

Figure 14
Estimate of the Viterbi inside score.

$$\text{Axiom}: \frac{}{0 : [S, \langle 0, len, 0\rangle]} \quad 1 \leq len \leq len_{max} \qquad \text{Unary}: \frac{out : [A, \vec{l}]}{out + log(p) : [B, \vec{l}]} \quad p : A(\vec{\alpha}) \rightarrow B(\vec{\alpha}) \in P$$

$$\text{Binary-right}: \frac{out : [X, \vec{l}_X]}{out + in(A, \vec{l}_A) + log(p) : [B, \vec{l}_B]}$$

$$\text{Binary-left}: \frac{out : [X, \vec{l}_X]}{out + in(B, \vec{l}_B) + log(p) : [A, \vec{l}_A]}$$

where, for both binary rules, there is an instantiated rule $p : X(\vec{\rho}) \rightarrow A(\vec{\rho_A})B(\vec{\rho_B})$ such that $\vec{l}_X = l_{out}(\rho), \vec{l}_A = l_{out}(\rho_A), \vec{l}_A = l_{in}(\rho_A), \vec{l}_B = l_{out}(\rho_B), \vec{l}_B = l_{in}(\rho_B)$.

Figure 15
Full SX estimate first version (top–down).

A item from $[B, \vec{l}]$ with probability p, then the log of the probability of $[A, \vec{l}]$ is greater or equal to $in + log(p)$. For each item, we record its maximal weight (i.e., its maximal probability). The rule **binary** is slightly more complicated because we have to compute the length vector of the left-hand side of the rule from the right-hand side length vectors.

A straightforward extension of the CFG algorithm from Klein and Manning (2003a) for computing the SX estimate is given in Figure 15. Here, the items have the form $[A, \vec{l}]$ where the vector $\vec{l}$ tells us about the lengths of the string to the left of the first component, the first component, the string in between the first and second component, and so on.

The algorithm proceeds top–down. The outside estimate of completing an S with component length len and no terminals to the left or to the right of the S component (item $[S, \langle 0, len, 0\rangle]$) is 0. If we expand with a unary rule (**unary**), then the outside estimate of the right-hand side item is greater or equal to the outside estimate of the left-hand side item plus the log of the probability of the rule. In the case of binary rules, we have to further add the inside estimate of the other daughter. For this, we need a different length vector (without the lengths of the parts in between the components). Therefore, for a given range vector $\rho = \langle\langle l_1, r_1\rangle, \ldots, \langle l_k, r_k\rangle\rangle$ and a sentence length n, we distinguish between the **inside length vector** $l_{in}(\rho) = \langle r_1 - l_1, \ldots, r_k - l_k\rangle$ and the **outside length vector** $l_{out}(\rho) = \langle l_1, r_1 - l_1, l_2 - r_1, \ldots, l_k - r_{k-1}, r_k - l_k, n - r_k\rangle$.

This algorithm has two major problems: Because it proceeds top–down, in the **binary** rules we must compute all splits of the antecedent X span into the spans of A and B, which is very expensive. Furthermore, for a category A with a certain number of terminals in the components and the gaps, we compute the lower part of the outside estimate several times, namely, for every combination of number of terminals to the left and to the right (first and last element in the outside length vector). In order to avoid these problems, we now abstract away from the lengths of the part to the left and the right, modifying our items such as to allow a bottom–up strategy.

The idea is to compute the weights of items representing the derivations from a certain lower C up to some A (C is a kind of "gap" in the yield of A) while summing up the inside costs of off-spine nodes and the log of the probabilities of the corresponding rules. We use items $[A, C, \rho_A, \rho_C, shift]$ where $A, C \in N$ and ρ_A, ρ_C are range vectors, both with a first component starting at position 0. The integer $shift \leq len_{max}$ tells us how many positions to the right the C span is shifted, compared to the starting position of the A. ρ_A and ρ_C represent the spans of C and A while disregarding the number of terminals to the left and the right (i.e., only the lengths of the components and of the gaps are encoded). This means in particular that the length n of the sentence does not play a role here. The right boundary of the last range in the vectors is limited to len_{max}. For

any $i, 0 \leq i \leq len_{max}$, and any range vector ρ, we define $shift(\rho, i)$ as the range vector one obtains from adding i to all range boundaries in ρ and $shift(\rho, -i)$ as the range vector one obtains from subtracting i from all boundaries in ρ.

The weight of $[A, C, \rho_A, \rho_C, i]$ estimates the log of the probability of completing a C tree with yield ρ_C into an A tree with yield ρ_A such that, if the span of A starts at position j, the span of C starts at position $i + j$. Figure 16 gives the computation. The value of $in(A, \vec{l})$ is the inside estimate of $[A, \vec{l}]$.

The SX-estimate for some predicate C with span ρ where i is the left boundary of the first component of ρ and with sentence length n is then given by the maximal weight of $[S, C, \langle 0, n \rangle, shift(\rho, -i), i]$.

4.2 SX with Left, Gaps, Right, Length

A problem of the previous estimate is that with a large number of non-terminals (for treebank parsing, approximately 12,000 after binarization and markovization), the computation of the estimate requires too much space. We therefore turn to simpler estimates with only a single non-terminal per item. We now estimate the outside score of a non-terminal A with a span of a length *length* (the sum of the lengths of all the components of the span), with *left* terminals to the left of the first component, *right* terminals to the right of the last component, and *gaps* terminals in between the components of the A span (i.e., filling the gaps). Our items have the form $[X, len, left, right, gaps]$ with $X \in N$, $len + left + right + gaps \leq len_{max}$, $len \geq dim(X)$, $gaps \geq dim(X) - 1$.

Let us assume that, in the rule $X(\vec{\alpha}) \rightarrow A(\vec{\alpha_A})B(\vec{\alpha_B})$, when looking at the vector $\vec{\alpha}$, we have $left_A$ variables for A-components preceding the first variable of a B component, $right_A$ variables for A-components following the last variable of a B component, and $right_B$ variables for B-components following the last variable of an A component. (In our grammars, the first left-hand side argument always starts with the first variable from A.) Furthermore, we set $gaps_A = dim(A) - left_A - right_A$ and $gaps_B = dim(B) - right_B$.

Figure 17 gives the computation of the estimate. It proceeds top–down, as the computation of the full SX estimate in Figure 15, except that now the items are simpler.

$$\text{POS tags: } \frac{}{0 : [C, C, \langle 0, 1 \rangle, \langle 0, 1 \rangle, 0]} \quad C \text{ a POS tag}$$

$$\text{Unary: } \frac{0 : [B, B, \rho_B, \rho_B, 0]}{log(p) : [A, B, \rho_B, \rho_B, 0]} \quad p : A(\vec{\alpha}) \rightarrow B(\vec{\alpha}) \in P$$

$$\text{Binary-right: } \frac{0 : [A, A, \rho_A, \rho_A, 0], 0 : [B, B, \rho_B, \rho_B, 0]}{in(A, l_{in}(\rho_A)) + log(p) : [X, B, \rho_X, \rho_B, i]}$$

$$\text{Binary-left: } \frac{0 : [A, A, \rho_A, \rho_A, 0], 0 : [B, B, \rho_B, \rho_B, 0]}{in(B, l_{in}(\rho_B)) + log(p) : [X, A, \rho_X, \rho_A, i]}$$

where i is such that for $shift(\rho_B, i) = \rho'_B$ $p : X(\rho_X) \rightarrow A(\rho_A)B(\rho'_B)$ is an instantiated rule.

$$\text{Starting sub-trees with larger gaps: } \frac{out : [B, C, \rho_B, \rho_C, i]}{0 : [B, B, \rho_B, \rho_B, 0]}$$

$$\text{Transitive closure of sub-tree combination: } \frac{out_1 : [A, B, \rho_A, \rho_B, i], out_2 : [B, C, \rho_B, \rho_C, j]}{out_1 + out_2 : [A, C, \rho_A, \rho_C, i + j]}$$

Figure 16
Full SX estimate second version (bottom–up).

$$\text{Axiom}: \frac{}{0 : [S, len, 0, 0, 0]} \quad 1 \leq len \leq len_{max}$$

$$\text{Unary}: \frac{out : [X, len, l, r, g]}{out + log(p) : [A, len, l, r, g]} \quad p : X(\vec{\alpha}) \to A(\vec{\alpha}) \in P$$

$$\text{Binary-right}: \frac{out : [X, len, l, r, g]}{out + in(A, len - len_B) + log(p) : [B, len_B, l_B, r_B, g_B]}$$

$$\text{Binary-left}: \frac{out : [X, len, l, r, g]}{out + in(B, len - len_A) + log(p) : [A, len_A, l_A, r_A, g_A]}$$

where, for both binary rules, $p : X(\vec{\alpha}) \to A(\vec{\alpha_A})B(\vec{\alpha_B}) \in P$.

Further side conditions for *Binary-right*:
a) $len + l + r + g = len_B + l_B + r_B + g_B$, b) $l_B \geq l + left_A$,
c) if $right_A > 0$, then $r_B \geq r + right_A$, else ($right_A = 0$), $r_B = r$, d) $g_B \geq gaps_A$.

Further side conditions for *Binary-left*:
a) $len + l + r + g = len_A + l_A + r_A + g_A$, b) $l_A = l$,
c) if $right_B > 0$, then $r_A \geq r + right_B$, else ($right_B = 0$), $r_A = r$ d) $g_A \geq gaps_B$.

Figure 17
SX estimate depending on length, left, right, gaps.

The value $in(X, l)$ for a non-terminal X and a length l, $0 \leq l \leq len_{max}$ is an estimate of the probability of an X category with a span of length l. Its computation is specified in Figure 18.

The SX-estimate for a sentence length n and for some predicate C with a range characterized by $\vec{\rho} = \langle\langle l_1, r_1\rangle, \ldots, \langle l_{dim(C)}, r_{dim(C)}\rangle\rangle$ where $len = \Sigma_{i=1}^{dim(C)}(r_i - l_i)$ and $r = n - r_{dim(C)}$ is then given by the maximal weight of the item $[C, len, l_1, r, n - len - l_1 - r]$.

4.3 SX with LR, Gaps, Length

In order to further decrease the space complexity of the computation of the outside estimate, we can simplify the previous estimate by subsuming the two lengths *left* and *right* in a single length *lr*. The items now have the form $[X, len, lr, gaps]$ with $X \in N$, $len + lr + gaps \leq len_{max}$, $len \geq dim(X)$, $gaps \geq dim(X) - 1$.

The computation is given in Figure 19. Again, we define $left_A$, $gaps_A$, $right_A$ and $gaps_B$, $right_B$ for a rule $X(\vec{\alpha}) \to A(\vec{\alpha_A})B(\vec{\alpha_B})$ as before. Furthermore, in both **Binary-left** and **Binary-right**, we have limited *lr* in the consequent item to the *lr* of the antecedent plus the length of the sister (len_B, resp. len_A). This results in a further reduction of the number of items while having only little effect on the parsing results.

The SX-estimate for a sentence length n and for some predicate C with a span $\vec{\rho} = \langle\langle l_1, r_1\rangle, \ldots, \langle l_{dim(C)}, r_{dim(C)}\rangle\rangle$ where $len = \Sigma_{i=1}^{dim(C)}(r_i - l_i)$ and $r = n - r_{dim(C)}$ is then the maximal weight of $[C, len, l_1 + r, n - len - l_1 - r]$.

$$\text{POS tags}: \frac{}{0 : [A, 1]} \quad A \text{ a POS tag} \qquad \text{Unary}: \frac{in : [B, l]}{in + log(p) : [A, l]} \quad p : A(\vec{\alpha}) \to B(\vec{\alpha}) \in P$$

$$\text{Binary}: \frac{in_B : [B, l_B], in_C : [C, l_C]}{in_B + in_C + log(p) : [A, l_B + l_C]}$$

where either $p : A(\vec{\alpha_A}) \to B(\vec{\alpha_B})C(\vec{\alpha_C}) \in P$ or $p : A(\vec{\alpha_A}) \to C(\vec{\alpha_C})B(\vec{\alpha_B}) \in P$.

Figure 18
Estimate of the inside score with total span length.

$$\text{Axiom}: \frac{}{0 : [S, len, 0, 0]} \quad 1 \leq len \leq len_{max}$$

$$\text{Unary}: \frac{out : [X, len, lr, g]}{out + log(p) : [A, len, lr, g]} \quad p : X(\vec{\alpha}) \rightarrow A(\vec{\alpha}) \in P$$

$$\text{Binary-right}: \frac{out : [X, len, lr, g]}{out + in(A, len - len_B) + log(p) : [B, len_B, lr_B, g_B]} \quad p : X(\vec{\alpha}) \rightarrow A(\vec{\alpha_A})B(\vec{\alpha_B}) \in P$$

$$\text{Binary-left}: \frac{out : [X, len, lr, g]}{out + in(B, len - len_A) + log(p) : [A, len_A, lr_A, g_A]} \quad p : X(\vec{\alpha}) \rightarrow A(\vec{\alpha_A})B(\vec{\alpha_B}) \in P$$

Further side conditions for *Binary-right*:
a) $len + lr + g = len_B + lr_B + g_B$ b) $lr < lr_B$ c) $g_B \geq gaps_A$
Further side conditions for *Binary-left*:
a) $len + lr + g = len_A + lr_A + g_A$ b) if $right_B = 0$ then $lr = lr_A$, else $lr < lr_A$ c) $g_A \geq gaps_B$

Figure 19
SX estimate depending on length, LR, gaps.

4.4 SX with Span and Sentence Length

We will now present a further simplification of the last estimate that records only the span length and the length of the entire sentence. The items have the form $[X, len, slen]$ with $X \in N$, $dim(X) \leq len \leq slen$. The computation is given in Figure 20. This last estimate is actually monotonic and allows for true A* parsing.

The SX-estimate for a sentence length n and for some predicate C with a span $\vec{\rho} = \langle\langle l_1, r_1\rangle, \ldots, \langle l_{dim(C)}, r_{dim(C)}\rangle\rangle$ where $len = \Sigma_{i=1}^{dim(C)}(r_i - l_i)$ is then the maximal weight of $[C, len, n]$.

In order to prove that this estimate allows for monotonic weighted deductive parsing and therefore guarantees that the best parse will be found, let us have a look at the CYK deduction rules when being augmented with the estimate. Only *Unary* and *Binary* are relevant because *Scan* does not have antecedent items. The two rules, augmented with the outside estimate, are shown in Figure 21.

We have to show that for every rule, if this rule has an antecedent item with weight w and a consequent item with weight w', then $w \geq w'$.

Let us start with *Unary*. To show: $in_B + out_B \geq in_B + log(p) + out_A$. Because of the *Unary* rule for computing the outside estimate and because of the unary production,

$$\text{Axiom}: \frac{}{0 : [S, len, len]} \quad 1 \leq len \leq len_{max}$$

$$\text{Unary}: \frac{out : [X, l_X, slen]}{out + log(p) : [A, l_X, slen]} \quad p : X(\vec{\alpha}) \rightarrow A(\vec{\alpha}) \in P$$

$$\text{Binary-right}: \frac{out : [X, l_X, slen]}{out + in(A, l_X - l_B) + log(p) : [B, l_B, slen]} \quad p : X(\vec{\alpha}) \rightarrow A(\vec{\alpha_A})B(\vec{\alpha_B}) \in P$$

$$\text{Binary-left}: \frac{out : [X, l_X, slen]}{out + in(B, l_X - l_A) + log(p) : [A, l_A, slen]} \quad p : X(\vec{\alpha}) \rightarrow A(\vec{\alpha_A})B(\vec{\alpha_B}) \in P$$

Figure 20
SX estimate depending on span and sentence length.

$$\text{Unary: } \frac{in_B + out_B : [B, \vec{\rho}]}{in_B + log(p) + out_A : [A, \vec{\rho}]} \quad p : A(\vec{\alpha}) \rightarrow B(\vec{\alpha}) \in P$$

$$\text{Binary: } \frac{in_B + out_B : [B, \vec{\rho_B}], in_C + out_C : [C, \vec{\rho_C}]}{in_B + in_C + log(p) + out_A : [A, \vec{\rho_A}]} \quad \begin{array}{l} p : A(\vec{\rho_A}) \rightarrow B(\vec{\rho_B})C(\vec{\rho_C}) \\ \text{is an instantiated rule} \end{array}$$

(Here, out_A, out_B, and out_C are the respective outside estimates of $[A, \vec{\rho_A}]$, $[B, \vec{\rho_B}]$ and $[C, \vec{\rho_C}]$.)

Figure 21
Parsing rules including outside estimate.

we obtain that, given the outside estimate out_A of $[A, \vec{\rho}]$ the outside estimate out_B of the item $[B, \vec{\rho}]$ is at least $out_A + log(p)$, namely, $out_B \geq log(p) + out_A$.

Now let us consider the rule *Binary*. We treat only the relation between the weight of the C antecedent item and the consequent. The treatment of the antecedent B is symmetric. To show: $in_C + out_C \geq in_B + in_C + log(p) + out_A$. Assume that l_B is the length of the components of the B item and n is the sentence length. Then, because of the *Binary-right* rule in the computation of the outside estimate and because of our instantiated rule $p : A(\vec{\rho_A}) \rightarrow B(\vec{\rho_B})C(\vec{\rho_C})$, we have that the outside estimate out_C of the C-item is at least $out_A + in(B, l_B) + log(p)$. Furthermore, $in(B, l_B) \geq in_B$. Consequently $out_C \geq in_B + log(p) + out_A$.

4.5 Integration into the Parser

Before parsing, the outside estimates of all items up to a certain maximal sentence length len_{max} are precomputed. Then, when performing the weighted deductive parsing as explained in Section 3.2, whenever a new item is stored in the agenda, we add its outside estimate to its weight.

Because the outside estimate is always greater than or equal to the actual outside score, given the input, the weight of an item in the agenda is always greater than or equal to the log of the actual product of the inside and outside score of the item. In this sense, the outside estimates given earlier are admissible.

Additionally, as already mentioned, note that the full SX estimate and the SX estimate with span and sentence length are monotonic and allow for A* parsing. The other two estimates, which are both not monotonic, act as FOMs in a best-first parsing context. Consequently, they contribute to speeding up parsing but they decrease the quality of the parsing output. For further evaluation details see Section 6.

5. Grammars for Discontinuous Constituents

5.1 Grammar Extraction

The algorithm we use for extracting an LCFRS from a constituency treebank with crossing branches has originally been presented in Maier and Søgaard (2008). It interprets the treebank trees as LCFRS derivation trees. Consider for instance the tree in Figure 22. The S node has two daughters, a VMFIN node and a VP node. This yields a rule $S \rightarrow VP\ VMFIN$. The VP is discontinuous with two components that wrap around the yield of the VMFIN. Consequently, the LCFRS rule is $S(XYZ) \rightarrow VP(X, Z)\ VMFIN(Y)$.

The extraction of an LCFRS from treebanks with crossing branches is almost immediate, except for the fan-out of the non-terminal categories: In the treebank, we can have the same non-terminal with different fan-outs, for instance a VP without a gap (fan-out 1), a VP with a single gap (fan-out 2), and so on. In the corresponding LCFRS,

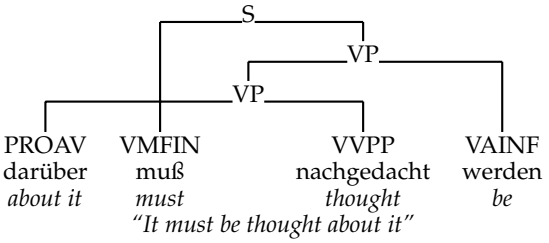


Figure 22
A sample tree from NeGra.

we have to distinguish these different non-terminals by mapping them to different predicates.

The algorithm first creates a so-called lexical clause $P(a) \rightarrow \varepsilon$ for each pre-terminal P dominating some terminal a. Then for all other non-terminals A_0 with the children $A_1 \cdots A_m$, a clause $A_0 \rightarrow A_1 \cdots A_m$ is created. The number of components of the $A_1 \cdots A_m$ is the number of discontinuous parts in their yields. The components of A_0 are concatenations of variables that describe how the discontinuous parts of the yield of A_0 are obtained from the yields of its daughters.

More precisely, the non-terminals in our LCFRS are all A_k where A is a non-terminal label in the treebank and k is a possible fan-out for A. For a given treebank tree $\langle V, E, r, l\rangle$ where V is the set of nodes, $E \subset V \times V$ the set of immediate dominance edges, $r \in V$ the root node, and $l : V \rightarrow N \cup T$ the labeling function, the algorithm constructs the following rules. Let us assume that $w_1, \ldots, w_n$ are the terminal labels of the leaves in $\langle V, E, r\rangle$ with a linear precedence relation $w_i \prec w_j$ for $1 \leq i < j \leq n$. We introduce a variable X_i for every w_i, $1 \leq i \leq n$.

- For every pair of nodes $v_1, v_2 \in V$ with $\langle v_2, v_2\rangle \in E$, $l(v_2) \in T$, we add $l(v_1)(l(v_2)) \rightarrow \varepsilon$ to the rules of the grammar. (We omit the fan-out subscript here because pre-terminals are always of fan-out 1.)
- For every node $v \in V$ with $l(v) = A_0 \notin T$ such that there are exactly m nodes $v_1, \ldots, v_m \in V$ ($m \geq 1$) with $\langle v, v_i\rangle \in E$ and $l(v_i) = A_i \notin T$ for all $1 \leq i \leq m$, we now create a rule

$$A_0(x_1^{(0)}, \ldots, x_{dim(A_0)}^{(0)}) \rightarrow A_1(x_1^{(1)}, \ldots, x_{dim(A_1)}^{(1)}) \ldots A_m(x_1^{(m)}, \ldots, x_{dim(A_m)}^{(m)})$$

  where for the predicate A_i, $0 \leq i \leq m$, the following must hold:

  1. The concatenation of all arguments of A_i, $x_1^{(i)} \ldots x_{dim(A_i)}^{(i)}$ is the concatenation of all $X \in \{X_i \mid \langle v_i, v_i'\rangle \in E^* \text{ with } l(v_i') = w_i\}$ such that X_i precedes X_j if $i < j$, and
  2. a variable X_j with $1 \leq j < n$ is the right boundary of an argument of A_i if and only if $X_{j+1} \notin \{X_i \mid \langle v_i, v_i'\rangle \in E^* \text{ with } l(v_i') = w_i\}$, that is, an argument boundary is introduced at each discontinuity.

  As a further step, in this new rule, all right-hand side arguments of length > 1 are replaced in both sides of the rule with a single new variable. Finally, all non-terminals A in the rule are equipped with an additional subscript $dim(A)$, which gives us the final non-terminal in our LCFRS.

$$
\begin{array}{rcl}
PROAV(\text{Darüber}) & \rightarrow & \varepsilon \\
VMFIN(\text{muß}) & \rightarrow & \varepsilon \\
VVPP(\text{nachgedacht}) & \rightarrow & \varepsilon \\
VAINF(\text{werden}) & \rightarrow & \varepsilon \\
S_1(X_1X_2X_3) & \rightarrow & VP_2(X_1, X_3)VMFIN(X_2) \\
VP_2(X_1, X_2X_3) & \rightarrow & VP_2(X_1, X_2)VAINF(X_3) \\
VP_2(X_1, X_2) & \rightarrow & PROAV(X_1)VVPP(X_2)
\end{array}
$$

Figure 23
LCFRS rules extracted from the tree in Figure 22.

For the tree in Figure 22, the algorithm produces for instance the rules in Figure 23.

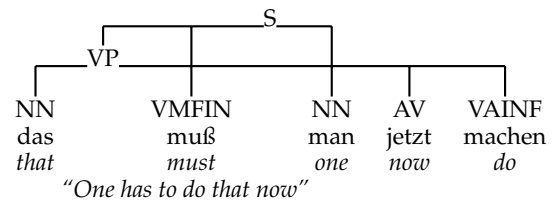
As standard for PCFG, the probabilities are computed using Maximum Likelihood Estimation.

5.2 Head-Outward Binarization

As previously mentioned, in contrast to CFG the order of the right-hand side elements of a rule does not matter for the result of an LCFRS derivation. Therefore, we can reorder the right-hand side of a rule before binarizing it.

The following, treebank-specific reordering results in a head-outward binarization where the head is the lowest subtree and it is extended by adding first all sisters to its left and then all sisters to its right. It consists of reordering the right-hand side of the rules extracted from the treebank such that first, all elements to the right of the head are listed in reverse order, then all elements to the left of the head in their original order, and then the head itself. Figure 24 shows the effect this reordering and binarization has on the form of the syntactic trees. In addition to this, we also use a variant of this reordering

Tree in NeGra format:



"One has to do that now"

Rule extracted for the S node: $S(XYZU) \rightarrow VP(X, U)VMFIN(Y)NN(Z)$
Reordering for head-outward binarization: $S(XYZU) \rightarrow NN(Z)VP(X, U)VMFIN(Y)$
New rules resulting from binarizing this rule:
$S(XYZ) \rightarrow S_{bin1}(X, Z)NN(Y) \quad S_{bin1}(XY, Z) \rightarrow VP(X, Z)VMFIN(Y)$
Rule extracted for the VP node: $VP(X, YZ) \rightarrow NN(X)AV(Y)VAINF(Z)$
New rules resulting from binarizing this rule:
$VP(X, Y) \rightarrow NN(X)VP_{bin1}(Y) \quad VP_{bin1}(XY) \rightarrow AV(X)VAINF(Y)$

Tree after binarization:

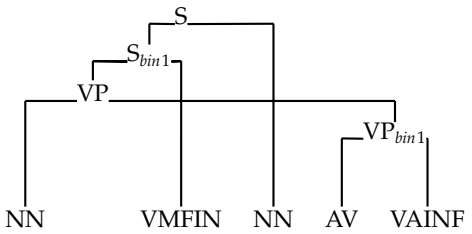


Figure 24
Sample head-outward binarization.

where we add first the sisters to the right and then the ones to the left. This is what Klein and Manning (2003b) do. To mark the heads of phrases, we use the head rules that the Stanford parser (Klein and Manning 2003c) uses for NeGra.

In all binarizations, there exists the possibility of adding additional unary rules when deriving the head. This allows for a further factorization. In the experiments, however, we do not insert unary rules, neither at the highest nor at the lowest new binarization non-terminal, because this was neither beneficial for parsing times nor for the parsing results.

5.3 Incorporating Additional Context

5.3.1 Markovization. As already mentioned in Section 3.1, a binarization that introduces unique new non-terminals for every single rule that needs to be binarized produces a large amount of non-terminals and fails to capture certain generalizations. For this reason, we introduce **markovization** (Collins 1999; Klein and Manning 2003b).

Markovization is achieved by introducing only a single new non-terminal for the new rules introduced during binarization and adding vertical and horizontal context from the original trees to each occurrence of this new non-terminal. As vertical context, we add the first v labels on the path from the root node of the tree that we want to binarize to the root of the entire treebank tree. The vertical context is collected during grammar extraction and then taken into account during binarization of the rules. As horizontal context, during binarization of a rule $A(\vec{\alpha}) \rightarrow A_0(\vec{\alpha_0}) \dots A_m(\vec{\alpha_m})$, for the new non-terminal that comprises the right-hand side elements $A_i \dots A_m$ (for some $1 \leq i \leq m$), we add the first h elements of $A_i, A_{i-1}, \dots, A_0$.

Figure 25 shows an example of a markovization of the tree from Figure 24 with $v = 1$ and $h = 2$. Here, the superscript is the vertical context and the subscript the horizontal context of the new non-terminal X. Note that in this example we have disregarded the fan-out of the context categories. The VP, for instance, is actually a VP_2 because it has fan-out 2. For the context symbols, one can either use the categories from the original treebank (without fan-out) or the ones from the LCFRS rules (with fan-out). We chose the latter approach because it delivered better parsing results.

5.3.2 Further Category Splitting. Grammar annotation (i.e., manual enhancement of annotation information through category splitting) has previously been successfully used in parsing German (Versley 2005). In order to see if such modifications can have a beneficial effect in PLCFRS parsing as well, we perform different category splits on the (unbinarized) NeGra constituency data.

We split the category S ("sentence") into SRC ("relative clause") and S (all other categories S). Relative clauses mostly occur in a very specific context, namely, as the

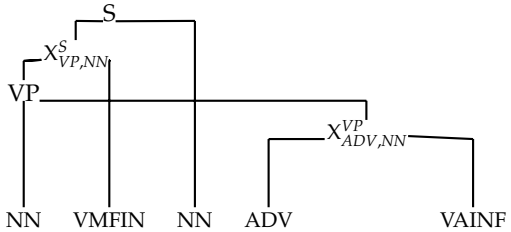


Figure 25
Sample markovization with $v = 1, h = 2$.

Table 1
NeGra: Properties of the data with crossing branches.

	training	test
number of sentences	16,502	1,833
average sentence length	14.56	14.62
average tree height	4.62	4.72
average children per node	2.96	2.94
sentences without gaps	12,481 (75.63%)	1,361 (74.25%)
sentences with one gap	3,320 (20.12%)	387 (21.11%)
sentences with $\geq$ 2 gaps	701 (4.25%)	85 (4.64%)
maximum gap degree	6	5

right part of an NP or a PP. This splitting should therefore speed up parsing and increase precision. Furthermore, we distinguish NPs by their case. More precisely, to all nodes with categories N, we append the grammatical function label to the category label. We finally experiment with the combination of both splits.

6. Experiments

6.1 Data

Our data source is the NeGra treebank (Skut et al. 1997). We create two different data sets for constituency parsing. For the first one, we start out with the unmodified NeGra treebank and remove all sentences with a length of more than 30 words. We pre-process the treebank following common practice (Kübler and Penn 2008), attaching all nodes which are attached to the virtual root node to nodes within the tree such that, ideally, no new crossing edges are created. In a second pass, we attach punctuation which comes in pairs (parentheses, quotation marks) to the same nodes. For the second data set we create a copy of the pre-processed first data set, in which we apply the usual tree transformations for NeGra PCFG parsing (i.e., moving nodes to higher positions until all crossing branches are resolved). The first 90% of both data sets are used as the training set and the remaining 10% as test set. The first data set is called NeGra_{LCFRS} and the second is called NeGra_{CFG}.

Table 1 lists some properties of the training and test (respectively, gold) parts of NeGra_{LCFRS}, namely, the total number of sentences, the average sentence length, the average tree height (the height of a tree being the length of the longest of all paths from the terminals to the root node), and the average number of children per node (excluding terminals). Furthermore, gap degrees (i.e., the number of gaps in the spans of non-terminal nodes) are listed (Maier and Lichte 2011).

Our findings correspond to those of Maier and Lichte except for small differences due to the fact that, unlike us, they removed the punctuation from the trees.

6.2 Parser Implementation

We have implemented the CYK parser described in the previous section in a system called `rparse`. The implementation is realized in Java.[3]

3 `rparse` is available under the GNU General Public License 2.0 at `http://www.phil.hhu.de/rparse`.

Table 2
NeGra$_{LCFRS}$: PLCFRS parsing results for different binarizations.

	Head-driven	KM	L-to-R	Optimal	Deterministic
LP	74.00	74.00	75.08	74.92	72.40
LR	74.24	74.13	74.69	74.88	71.80
LF_1	74.12	74.07	74.88	74.90	72.10
UP	77.09	77.20	77.95	77.77	75.67
UR	77.34	77.33	77.54	77.73	75.04
UF_1	77.22	77.26	77.75	77.75	75.35

6.3 Evaluation

For the evaluation of the constituency parses, we use an EVALB-style metric. For a tree over a string w, a single constituency is represented by a tuple $\langle A, \vec{\rho} \rangle$ with A being a node label and $\vec{\rho} \in (Pos(w) \times Pos(w))^{dim(A)}$. We compute precision, recall, and F_1 based on these tuples from gold and de-binarized parsed test data from which all category splits have been removed. This metric is equivalent to the corresponding PCFG metric for $dim(A) = 1$. Despite the shortcomings of such a measure (Rehbein and van Genabith 2007), it still allows to some extent a comparison to previous work in PCFG parsing (see also Section 7). Note that we provide the parser with gold POS tags in all experiments.

6.4 Markovization and Binarization

We use the markovization settings $v = 1$ and $h = 2$ for all further experiments. The setting which has been reported to yield the best results for PCFG parsing of NeGra, $v = 2$ and $h = 1$ (Rafferty and Manning 2008), required a parsing time which was too high.[4]

Table 2 contains the parsing results for NeGra$_{LCFRS}$ using five different binarizations: *Head-driven* and *KM* are the two head-outward binarizations that use a head chosen on linguistic grounds (described in Section 5.2); *L-to-R* is another variant in which we always choose the rightmost daughter of a node as its head.[5] *Optimal* reorders the left-hand side such that the fan-out of the binarized rules is optimized (described in Section 3.1.2). Finally, we also try a deterministic binarization (*Deterministic*) in which we binarize strictly from left to right (i.e., we do not reorder the right-hand sides of productions, and choose unique binarization labels).

The results of the head-driven binarizations and the optimal binarization lie close together; the results for the deterministic binarization are worse. This indicates that the presence or absence of markovization has more impact on parsing results than the actual binarization order. Furthermore, the non-optimal binarizations did not yield a binarized grammar of a higher fan-out than the optimal binarization: For all five binarizations, the fan-out was 7 (caused by a VP interrupted by punctuation).

4 Older versions of `rparse` contained a bug that kept the priority queue from being updated correctly (i.e., during an update, the corresponding node in the priority queue was not moved to its top, and therefore the best parse was not guaranteed to be found); however, higher parsing speeds were achieved. The current version of `rparse` implements the update operation correctly, using a Fibonacci queue to ensure efficiency (Cormen et al. 2003). Thanks to Andreas van Cranenburgh for pointing this out.

5 The term *head* is not used in its proper linguistic sense here.

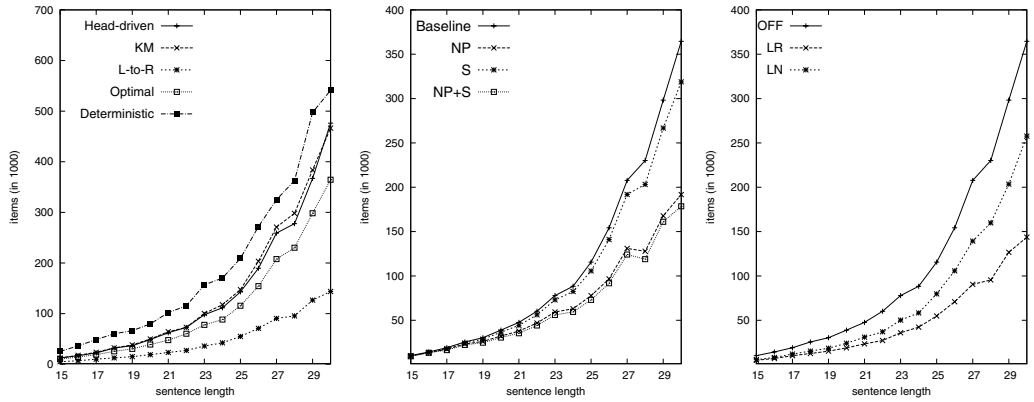


Figure 26
NeGra$_{LCFRS}$: Items for PLCFRS parsing (left-to-right): binarizations, baseline and category splits, and estimates.

The different binarizations result in different numbers of items, and therefore allow for different parsing speeds. The respective leftmost graph in Figures 26 and 27 show a visual representation of the number of items produced by all binarizations, and the corresponding parsing times. Note that when choosing the head with head rules the number of items is almost not affected by the choice of adding first the children to the left of the head and then to the right of the head or vice versa. The optimal binarization produces the best results. Therefore we will use it in all further experiments, in spite of its higher parsing time.

6.5 Baseline Evaluation and Category Splits

Table 3 presents the constituency parsing results for NeGra$_{LCFRS}$ and NeGra$_{CFG}$, both with and without the different category splits. Recall that NeGra$_{LCFRS}$ has crossing branches and consequently leads to a PLCFRS of fan-out > 1 whereas NeGra$_{CFG}$ does not contain crossing branches and consequently leads to a 1-PLCFRS—in other words,

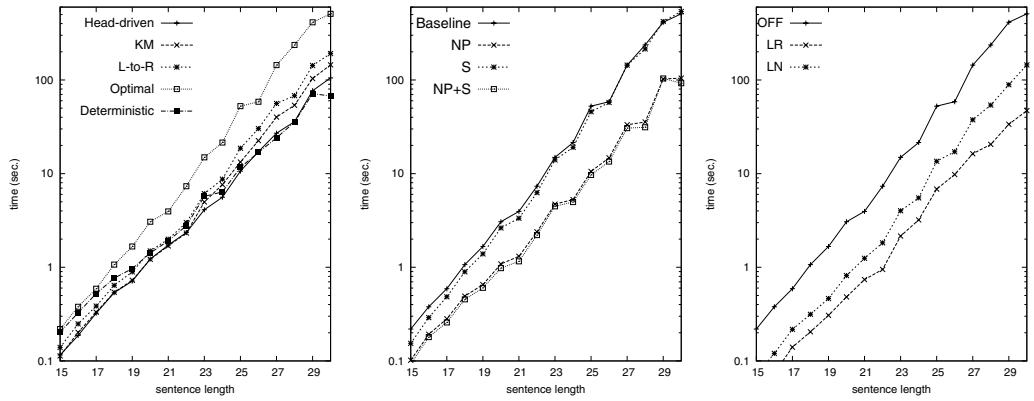


Figure 27
NeGra$_{LCFRS}$: Parsing times for PLCFRS parsing (left-to-right): binarizations, baseline and category splits, and estimates (log scale).

Table 3
NeGra_{LCFRS} and NeGra_{CFG}: baseline and category splits.

		w/ category splits				w/ category splits		
	NeGra_{LCFRS}	*NP*	*S*	*NP* ∘ *S*	NeGra_{CFG}	*NP*	*S*	*NP* ∘ *S*
LP	74.92	75.21	75.81	75.93	76.32	76.79	77.39	77.58
LR	74.88	74.95	75.65	75.57	76.36	77.23	77.35	77.99
LF_1	74.90	75.08	75.73	75.75	76.34	77.01	77.37	77.79
UP	77.77	78.16	78.31	78.60	79.12	79.62	79.84	80.09
UR	78.73	77.88	78.15	78.22	79.17	80.08	79.80	80.52
UF_1	77.75	78.02	78.23	78.41	79.14	79.85	79.82	80.30

a PCFG. We evaluate the parser output against the unmodified gold data; that is, before we evaluate the experiments with category splits, we replace all split labels in the parser output with the corresponding original labels.

We take a closer look at the properties of the trees in the parser output for NeGra_{LCFRS}. Twenty-nine sentences had no parse, therefore, the parser output has 1,804 sentences. The average tree height is 4.72, and the average number of children per node (excluding terminals) is 2.91. These values are almost identical to the values for the gold data. As for the gap degree, we get 1,401 sentences with no gaps (1,361 in the gold set), 334 with gap degree 1 (387 in the gold set), and 69 with 2 or 3 gaps (85 in the gold set). Even though the difference is only small, one can see that fewer gaps are preferred. This is not surprising, since constituents with many gaps are rare events and therefore end up with a probability which is too low.

We see that the quality of the PLCFRS parser output on NeGra_{LCFRS} (which contains more information than the output of a PCFG parser) does not lag far behind the quality of the PCFG parsing results on NeGra_{CFG}. With respect to the category splits, the results show furthermore that category splitting is indeed beneficial for the quality of the PLCFRS parser output. The gains in speed are particularly visible for sentences with a length greater than 20 words (cf. the number of produced items and parsing times in Figures 26 and 27 [middle]).

6.6 Evaluating Outside Estimates

We compare the parser performance without estimates (OFF) with its performance with the estimates described in Sections 4.3 (LR) and 4.4 (LN).

Unfortunately, the full estimates seem to be only of theoretical interest because they were too expensive to compute both in terms of time and space, given the restrictions imposed by our hardware. We could, however, compute the LN and the LR estimate. Unlike the LN estimate, which allows for true A^* parsing, the LR estimate lets the quality of the parsing results deteriorate: Compared with the baseline, labeled F_1 drops from 74.90 to 73.76 and unlabeled F_1 drops from 77.91 to 76.89. The respective rightmost graphs in Figures 26 and 27 show the average number of items produced by the parser and the parsing times for different sentence lengths. The results indicate that the estimates have the desired effect of preventing unnecessary items from being produced. This is reflected in a significantly lower parsing time.

The different behavior of the LR and the LN estimate raises the question of the trade-off between maintaining optimality and obtaining a higher parsing speed. In

other words, it raises the question of whether techniques such as pruning or coarse-to-fine parsing (Charniak et al. 2006) would probably be superior to A* parsing. A first implementation of a coarse-to-fine approach has been presented by van Cranenburgh (2012). He generates a CFG from the treebank PLCFRS, based on the idea of Barthélemy et al. (2001). This grammar, which can be seen as a coarser version of the actual PLCFRS, is then used for pruning of the search space. The problem that van Cranenburgh tackles is specific to PLCFRS: His PCFG stage generalizes over the distinction of labels by their fan-out. The merit of his work is an enormous increase in efficiency: Sentences with a length of up to 40 words can now be parsed in a reasonable time. For a comparison of the results of van Cranenburgh (2012) with our work, the same version of evaluation parameters would have to be used. The applicability and effectiveness of other coarse-to-fine approaches (Charniak et al. 2006; Petrov and Klein 2007) on PLCFRS remain to be seen.

7. Comparison to Other Approaches

Comparing our results with results from the literature is a difficult endeavor, because PLCFRS parsing of NeGra is an entirely new task that has no direct equivalent in previous work. In particular, it is a harder task than PCFG parsing. What we can provide in this section is a comparison of the performance of our parser on NeGra_{CFG} to the performance of previously presented PCFG parsers on the same data set and an overview on previous work on parsing which aims at reconstructing crossing branches.

For the comparison of the performance of our parser on NeGra_{CFG}, we have performed experiments with Helmut Schmid's LoPar (Schmid 2000) and with the Stanford Parser (Klein and Manning 2003c) on NeGra_{CFG}.[6] For the experiments both parsers were provided with gold POS tags. Recall that our parser produced labeled precision, recall, and F_1 of 76.32, 76.46, and 76.34, respectively. The plain PCFG provided by LoPar delivers lower results (LP 72.86, LR 74.43, and LF_1 73.63). The Stanford Parser results (markovization setting $v = 2$, $h = 1$ [Rafferty and Manning 2008], otherwise default parameters) lie in the vicinity of the results of our parser (LP 74.27, LR 76.19, LF_1 75.45). Although the results for LoPar are no surprise, given the similarity of the models implemented by our parser and the Stanford parser, it remains to be investigated why the lexicalization component of the Stanford parser does not lead to better results. In any case the comparison shows that on a data set without crossing branches, our parser obtains the results one would expect. A further data set to which we can provide a comparison is the PaGe workshop experimental data (Kübler and Penn 2008).[7] Table 4 lists the results of some of the papers in Kübler and Penn (2008) on TIGER, namely, for Petrov and Klein (2008) (P&K), who use the Berkeley Parser (Petrov and Klein 2007); Rafferty and Manning (2008) (R&M), who use the Stanford parser (see above); and Hall and Nivre (2008) (H&N), who use a dependency-based approach (see next paragraph). The comparison again shows that our system produces good results. Again the performance gap between the Stanford parser and our parser warrants further investigation.

6 We have obtained the former parser from `http://www.ims.uni-stuttgart.de/tcl/SOFTWARE/LoPar.html` and the latter (Version 2.0.1) from `http://nlp.stanford.edu/software/lex-parser.shtml`.

7 Thanks to Sandra Kübler for providing us with the experimental data.

Table 4
PaGe workshop data.

	here	P&K	R&M	H&N
LP	66.93	69.23	58.52	67.06
LR	60.79	70.41	57.63	58.07
LF_1	63.71	69.81	58.07	65.18

As for the work that aims to create crossing branches, Plaehn (2004) obtains 73.16 Labeled F_1 using Probabilistic Discontinuous Phrase Structure Grammar (DPSG), albeit only on sentences with a length of up to 15 words. On those sentences, we obtain 83.97. The crucial difference between DPSG rules and LCFRS rules is that the former explicitly specify the material that can occur in gaps whereas LCFRS does not. Levy (2005), like us, proposes to use LCFRS but does not provide any evaluation results of his work. Very recently, Evang and Kallmeyer (2011) followed up on our work. They transform the Penn Treebank such that the trace nodes and co-indexations are converted into crossing branches and parse them with the parser presented in this article, obtaining promising results. Furthermore, van Cranenburgh, Scha, and Sangati (2011) and van Cranenburgh (2012) have also followed up on our work, introducing an integration of our approach with Data-Oriented Parsing (DOP). The former article introduces an LCFRS adaption of Goodman's PCFG-DOP (Goodman 2003). For their evaluation, the authors use the same data as we do in Maier (2010), and obtain an improvement of roughly 1.5 points F-measure. They are also confronted with the same efficiency issues, however, and encounter a bottleneck in terms of parsing time. In van Cranenburgh (2012), a coarse-to-fine approach is presented (see Section 6.6). With this approach much faster parsing is made possible and sentences with a length of up to 40 words can be parsed. The cost of the speed, however, is that the results lie well below the baseline results for standard PLCFRS parsing.

A comparison with non-projective dependency parsers (McDonald et al. 2005; Nivre et al. 2007) might be interesting as well, given that non-projectivity is the dependency-counterpart to discontinuity in constituency parsing. A meaningful comparison is difficult to do for the following reasons, however. Firstly, dependency parsing deals with relations between words, whereas in our case words are not considered in the parsing task. Our grammars take POS tags for a given and construct syntactic trees. Also, dependency conversion algorithms generally depend on the correct identification of linguistic head words (Lin 1995). We cannot rely on grammatical function labels, such as, for example, Boyd and Meurers (2008). Therefore we would have to use heuristics for the dependency conversion of the parser output. This would introduce additional noise. Secondly, the resources one obtains from our PLCFRS parser and from dependency parsers (the probabilistic LCFRS and the trained dependency parser) are quite different because the former contains non-lexicalized internal phrase structure identifying meaningful syntactic categories such as VP or NP while the latter is only concerned with relations between lexical items. A comparison would concentrate only on relations between lexical items and the rich phrase structure provided by a constituency parser would not be taken into account. To achieve some comparison, one could of course transform the discontinuous constituency trees into dependency trees with dependencies between heads and with edge labels that encode enough of the syntactic structure to retrieve the original constituency tree (Hall and Nivre 2008). The result could then be used for

a dependency evaluation. It is not clear what is to gain by this evaluation because the head-to-head dependencies one would obtain are not necessarily the predicate-argument dependencies one would aim at when doing direct dependency parsing (Rambow 2010).[8]

8. Conclusion

We have presented the first efficient implementation of a weighted deductive CYK parser for Probabilistic Linear Context-Free Rewriting Systems (PLCFRS), showing that LCFRS indeed allows for data-driven parsing while modeling discontinuities in a straightforward way. To speed up parsing, we have introduced different context-summary estimates of parse items, some acting as figures-of-merit, others allowing for A* parsing. We have implemented the parser and we have evaluated it with grammars extracted from the German NeGra treebank. Our experiments show that data-driven LCFRS parsing is feasible and yields output of competitive quality.

There are three main directions for future work on this subject.

- On the symbolic side, LCFRS seems to offer more power than necessary. By removing symbolic expressivity, a lower parsing complexity can be achieved. One possibility is to disallow the use of so-called ill-nested LCFRS rules. These are rules where, roughly, the spans of two right-hand side non-terminals interleave in a cross-serial way. See the parsing algorithm in Gómez-Rodríguez, Kuhlmann, and Satta (2010). Nevertheless, this seems to be too restrictive for linguistic modeling (Chen-Main and Joshi 2010; Maier and Lichte 2011). Our goal for future work is therefore to define reduced forms of ill-nested rules with which we get a lower parsing complexity.
  Another possibility is to reduce the fan-out of the extracted grammar. We have pursued the question whether the fan-out of the trees in the treebank can be reduced in a linguistically meaningful way in Maier, Kaeshammer, and Kallmeyer (2012).
- On the side of the probabilistic model, there are certain independence assumptions made in our model that are too strong. The main problem in respect is that, due to the definition of LCFRS, we have to distinguish between occurrences of the same category with different fan-outs. For instance, VP_1 (no gaps), VP_2 (one gap), and so on, are different non-terminals. Consequently, the way they expand are considered independent from each other. This is of course not true, however. Furthermore, some of these non-terminals are rather rare; we therefore have a sparse data problem here. This leads to the idea to separate the development of a category (independent from its fan-out) and the fan-out and position of gaps. We plan to integrate this into our probabilistic model in future work.

8 A way to overcome this difference in the content of the dependency annotation would be to use an evaluation along the lines of Tsarfaty, Nivre, and Andersson (2011); this is not available yet for annotations with crossing branches, however.

- Last, it is clear that a more informative evaluation of the parser output is still necessary, particularly with respect to its performance at the task of finding long distance dependencies and with respect to its behavior when not provided with gold POS tags.

Acknowledgments

We are particularly grateful to Giorgio Satta for extensive discussions of the details of the probabilistic treebank model presented in this paper. Furthermore, we owe a debt to Kilian Evang who participated in the implementation of the parser. Thanks to Andreas van Cranenburgh for helpful feedback on the parser implementation. Finally, we are grateful to our three anonymous reviewers for many valuable and helpful comments and suggestions. A part of the work on this paper was funded by the German Research Foundation DFG (Deutsche Forschungsgemeinschaft) in the form of an Emmy Noether Grant and a subsequent DFG research project.

References

Barthélemy, François, Pierre Boullier, Philippe Deschamp, and Éric Villemonte de la Clergerie. 2001. Guided parsing of range concatenation languages. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 42–49, Toulouse.

Becker, Tilman, Aravind K. Joshi, and Owen Rambow. 1991. Long-distance scrambling and tree-adjoining grammars. In *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–26, Berlin.

Boullier, Pierre. 1998a. A generalization of mildly context-sensitive formalisms. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+4)*, pages 17–20, Philadelphia, PA.

Boullier, Pierre. 1998b. A proposal for a natural language processing syntactic backbone. Technical Report 3342, INRIA, Roquencourt.

Boullier, Pierre. 2000. Range concatenation grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT2000)*, pages 53–64, Trento.

Boyd, Adriane. 2007. Discontinuity revisited: An improved conversion to context-free representations. In the *Linguistic Annotation Workshop at ACL 2007*, pages 41–44, Prague.

Boyd, Adriane and Detmar Meurers. 2008. Revisiting the impact of different annotation schemes on PCFG parsing: A grammatical dependency evaluation. In *Proceedings of the Workshop on Parsing German at ACL 2008*, pages 24–32, Columbus, OH.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories*, pages 24–42, Sozopol.

Bunt, Harry. 1996. Formal tools for describing and processing discontinuous constituency structure. In Harry Bunt and Arthur van Horck, editors, *Discontinuous Constituency*, volume 6 of *Natural Language Processing*. Mouton de Gruyter, Berlin, pages 63–83.

Cai, Shu, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 212–216, Portland, OR.

Candito, Marie and Djamé Seddah. 2010. Parsing word clusters. In *Proceedings of the First Workshop on Statistical Parsing of Morphologically-Rich Languages at NAACL HLT 2010*, pages 76–84, Los Angeles, CA.

Caraballo, Sharon A. and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298.

Charniak, Eugene, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. 2006. Multilevel coarse-to-fine PCFG parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 168–175, New York, NY.

Chen-Main, Joan and Aravind Joshi. 2010. Unavoidable ill-nestedness in natural language and the adequacy of tree local-MCTAG induced dependency structures. In *Proceedings of the Tenth*

International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+10), pages 119–126, New Haven, CT.

Chiang, David. 2003. Statistical parsing with an automatically extracted tree adjoining grammar. In Rens Bod, Remko Scha, and Khalil Sima'an, editors, *Data-Oriented Parsing*, CSLI Studies in Computational Linguistics. CSLI Publications, Stanford, CA, pages 299–316.

Collins, Michael. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2003. *Introduction to Algorithms*. MIT Press, Cambridge, 2nd edition.

Dienes, Péter. 2003. *Statistical Parsing with Non-local Dependencies*. Ph.D. thesis, Saarland University.

Evang, Kilian and Laura Kallmeyer. 2011. PLCFRS parsing of English discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116, Dublin.

Gómez-Rodríguez, Carlos, Marco Kuhlmann, and Giorgio Satta. 2010. Efficient parsing of well-nested linear context-free rewriting systems. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 276–284, Los Angeles, CA.

Gómez-Rodríguez, Carlos, Marco Kuhlmann, Giorgio Satta, and David Weir. 2009. Optimal reduction of rule length in linear context-free rewriting systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 539–547, Boulder, CO.

Goodman, Joshua. 2003. Efficient parsing of DOP with PCFG-reductions. In Rens Bod, Remko Scha, and Khalil Sima'an, editors, *Data-Oriented Parsing*, CSLI Studies in Computational Linguistics. CSLI Publications, Stanford, CA, pages 125–146.

Hall, Johan and Joakim Nivre. 2008. A dependency-driven parser for German dependency and constituency representations. In *Proceedings of the Workshop on Parsing German at ACL 2008*, pages 47–54, Columbus, OH.

Han, Chung-hye, Na-Rae Han, and Eon-Suk Ko. 2001. Bracketing guidelines for Penn Korean TreeBank. Technical Report 01-10, IRCS, University of Pennsylvania, Philadelphia, PA.

Hockenmaier, Julia. 2003. *Data and models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Hoffman, Beryl. 1995. Integrating "free" word order syntax and information structure. In *Seventh Conference of the European Chapter of the Association for Computational Linguistics*, pages 245–251, Dublin.

Höhle, Tilman. 1986. Der Begriff "Mittelfeld"—Anmerkungen über die Theorie der topologischen Felder. In *Akten des Siebten Internationalen Germanistenkongresses 1985*, Göttingen, Germany.

Johnson, Mark. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143, Philadelphia, PA.

Kallmeyer, Laura. 2010. *Parsing Beyond Context-Free Grammars*. Springer, Berlin.

Kallmeyer, Laura and Wolfgang Maier. 2010. Data-driven parsing with probabilistic linear context-free rewriting systems. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 537–545, Beijing.

Kato, Yuki, Hiroyuki Seki, and Tadao Kasami. 2006. Stochastic multiple context-free grammar for RNA pseudoknot modeling. In *Proceedings of* the *Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+8)*, pages 57–64, Sydney.

Klein, Dan and Christopher D. Manning. 2003a. A* Parsing: Fast exact viterbi parse selection. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 40–47, Edmonton.

Klein, Dan and Christopher D. Manning. 2003b. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo.

Klein, Dan and Christopher D. Manning. 2003c. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 3–10, Vancouver.

Kracht, Marcus. 2003. *The Mathematics of Language*. Number 63 in *Studies in Generative Grammar*. Mouton de Gruyter, Berlin.

Kübler, Sandra. 2005. How do treebank annotation schemes influence parsing results? Or how not to compare apples and oranges. In *Recent Advances in Natural Language Processing 2005 (RANLP 2005)*, pages 293–300, Borovets.

Kübler, Sandra and Gerald Penn, editors. 2008. *Proceedings of the Workshop on Parsing German at ACL 2008*. Association for Computational Linguistics, Columbus, OH.

Kuhlmann, Marco and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 478–486, Athens.

Levy, Roger. 2005. *Probabilistic Models of Word Order and Syntactic Discontinuity*. Ph.D. thesis, Stanford University.

Levy, Roger and Christopher D. Manning. 2004. Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 328–335, Barcelona.

Lin, Dekang. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, pages 1420–1427, Montreal.

Maier, Wolfgang. 2010. Direct parsing of discontinuous constituents in German. In *Proceedings of the First Workshop on Statistical Parsing of Morphologically-Rich Languages at NAACL HLT 2010*, pages 58–66, Los Angeles, CA.

Maier, Wolfgang, Miriam Kaeshammer, and Laura Kallmeyer. 2012. Data-driven PLCFRS parsing revisited: Restricting the fan-out to two. In *Proceedings of the Eleventh International Conference on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pages 126–134, Paris.

Maier, Wolfgang and Laura Kallmeyer. 2010. Discontinuity and non-projectivity: Using mildly context-sensitive formalisms for data-driven parsing. In *Proceedings of the Tenth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+10)*, New Haven, CT.

Maier, Wolfgang and Timm Lichte. 2011. Characterizing discontinuity in constituent treebanks. In *Formal Grammar. 14th International Conference, FG 2009. Bordeaux, France, July 25-26, 2009. Revised Selected Papers*, volume 5591 of *Lecture Notes in Artificial Intelligence*, pages 167–182, Springer-Verlag, Berlin/Heidelberg/New York.

Maier, Wolfgang and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In *Proceedings of the 13th Conference on Formal Grammar (FG-2008)*, pages 61–76, Hamburg.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Human Language Technology Conference*, pages 114–119.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530, Vancouver.

Michaelis, Jens. 2001. *On Formal Properties of Minimalist Grammars*. Ph.D. thesis, Universität Potsdam.

Müller, Gereon. 2002. Free word order, morphological case, and sympathy theory. In Gisbert Fanselow and Caroline Fery, editors, *Resolving Conflicts in Grammars: Optimality Theory in Syntax, Morphology, and Phonology*. Buske Verlag, Hamburg, pages 265–397.

Müller, Stefan. 2004. Continuous or discontinuous constituents? *Research on Language & Computation*, 2(2):209–257.

Nederhof, Mark-Jan. 2003. Weighted deductive parsing and knuth's algorithm. *Computational Linguistics*, 29(1):135–143.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Osenova, Petya and Kiril Simov. 2004. BTB-TR05: BulTreebank Stylebook. Technical Report 05, BulTreeBank Project, Sofia, Bulgaria.

Pereira, Fernando C. N. and David Warren. 1983. Parsing as deduction. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 137–144, Cambridge, MA.

Petrov, Slav and Dan Klein. 2007. Improved inference for unlexicalized parsing.

In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, NY.

Petrov, Slav and Dan Klein. 2008. Parsing German with latent variable grammars. In *Proceedings of the Workshop on Parsing German at ACL 2008*, pages 24–32, Columbus, OH.

Plaehn, Oliver. 2004. Computing the most probable parse for a discontinuous phrase-structure grammar. In Harry Bunt, John Carroll, and Giorgio Satta, editors, *New Developments in Parsing Technology*, volume 23 of *Text, Speech And Language Technology*. Kluwer, Dordrecht, pages 91–106.

Rafferty, Anna and Christopher D. Manning. 2008. Parsing three German treebanks: Lexicalized and unlexicalized baselines. In *Proceedings of the Workshop on Parsing German at ACL 2008*, pages 40–46, Columbus, OH.

Rambow, Owen. 2010. The simple truth about dependency and phrase structure representations: An opinion piece. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 337–340, Los Angeles, CA.

Rehbein, Ines and Josef van Genabith. 2007. Evaluating evaluation measures. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, pages 372–379, Tartu.

Schmid, Helmut. 2000. LoPar: Design and implementation. Arbeitspapiere des Sonderforschungsbereiches 340 149, IMS, University of Stuttgart, Stuttgart, Germany.

Seddah, Djame, Sandra Kübler, and Reut Tsarfaty, editors. 2010. *Proceedings of the First Workshop on Statistical Parsing of Morphologically-Rich Languages at NAACL HLT 2010*. Association for Computational Linguistics, Los Angeles, CA.

Seki, Hiroyuki, Takahashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36.

Sikkel, Klaas. 1997. *Parsing Schemata*. Texts in Theoretical Computer Science. Springer, Berlin, Heidelberg, New York.

Skut, Wojciech, Brigitte Krenn, Thorten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP)*, pages 88–95, Washington, DC.

Telljohann, Heike, Erhard W. Hinrichs, Sandra Kübler, Heike Zinsmeister, and Kathrin Beck. 2012. Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z). Technical report, Seminar für Sprachwissenschaft, Universität Tübingen, Tübingen, Germany. `http://www.sfs.uni.tuebingen.de/resources/tuebadz-stylebook-1201.pdf`.

Tsarfaty, Reut, Joakim Nivre, and Evelina Andersson. 2011. Evaluating dependency parsing: Robust and heuristics-free cross-annotation evaluation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 385–396, Edinburgh.

Uszkoreit, Hans. 1986. Linear precedence in discontinuous constituents: Complex fronting in German. CSLI report CSLI-86-47, Center for the Study of Language and Information, Stanford University, Stanford, CA.

van Cranenburgh, Andreas. 2012. Efficient parsing with linear context-free rewriting systems. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 460–470, Avignon.

van Cranenburgh, Andreas, Remko Scha, and Federico Sangati. 2011. Discontinuous data-oriented parsing: A mildly context-sensitive all-fragments grammar. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2011)*, pages 34–44, Dublin.

Versley, Yannick. 2005. Parser evaluation across text types. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories*, pages 209–220, Barcelona, Spain.

Vijay-Shanker, K., David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Stanford, CA.