PLCFRS Parsing Revisited: Restricting the Fan-Out to Two

Wolfgang Maier, **Miriam Kaeshammer** and **Laura Kallmeyer**
Institut für Sprache und Information
University of Düsseldorf
Universitätsstr. 1, 40225 Düsseldorf, Germany
{maierwo,kaeshammer,kallmeyer}@phil.hhu.de

Abstract

Linear Context-Free Rewriting System (LCFRS) is an extension of Context-Free Grammar (CFG) in which a non-terminal can dominate more than a single continuous span of terminals. Probabilistic LCFRS have recently successfully been used for the direct data-driven parsing of discontinuous structures. In this paper we present a parser for binary PLCFRS of fan-out two, together with a novel monotonous estimate for A* parsing, with which we conduct experiments on modified versions of the German NeGra treebank and the Discontinuous Penn Treebank in which all trees have block degree two. The experiments show that compared to previous work, our approach provides an enormous speed-up while delivering an output of comparable richness.

1 Introduction

In many constituency treebanks, the syntactic annotation takes the form of Context-Free Grammar (CFG) derivation trees, i.e., of trees with no crossing branches. Discontinuous structures (Huck and Ojeda, 1987) cannot be modeled with CFG and are therefore handled by an additional mechanism in such an annotation. In the Penn Treebank (PTB) (Marcus et al., 1993), for instance, a combination of trace nodes and co-indexation labels is used in order to establish implicit edges. In other treebanks, e.g., the German NeGra (Skut et al., 1997) and TIGER (Brants et al., 2002) treebanks, crossing branches are allowed.[1] This way, all parts of a discontinuous constituent can be grouped under a single node. There is no fundamental difference between both representations: PTB-style annotation can be converted into a NeGra/TIGER-style annotation. This has been done in the Discontinuous Penn Treebank (DPTB) (Evang and Kallmeyer, 2011).

For data-driven parsing with Probabilistic CFG (PCFG), the annotation information concerning discontinuities must be discarded, because it exceeds the expressivity of CFG. For NeGra, there exist two methods, namely (i) attaching non-head daughters of discontinuous constituents to higher positions in the tree, such that the crossing branches disappear (the NeGra distribution contains a version of the treebank in which this transformation is readily carried out), or (ii) introducing an additional non-terminal node for each continuous part of a discontinuous constituent (Boyd, 2007). As an example, figure 1 shows the annotation of (1) before and after both transformations.

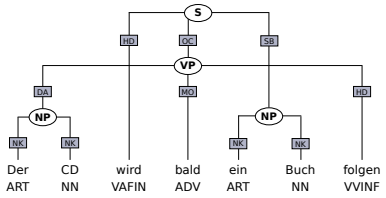
(1) Der CD wird bald ein Buch folgen
The CD will soon a book follow
"Soon, the CD will be followed by a book."

For PCFG parsing with the PTB, trace nodes and co-indexation are simply discarded. With either of these transformations, discontinuities are lost and cannot be restored from the parser output. However, the fact that about 25%, resp. 20% of all sentences in NeGra, resp. the PTB contain discontinuities (Maier and Lichte, 2011; Evang and Kallmeyer, 2011) shows that this is an undesirable situation and that these structures warrant a proper treatment.

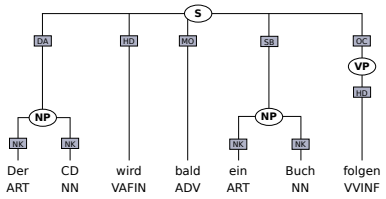
Linear Context-Free Rewriting System (LCFRS), an extension of CFG, has been established as an appropriate candidate for modeling

[1] The annotation differences between TIGER and NeGra are minor and can be neglected for the purpose of this work.

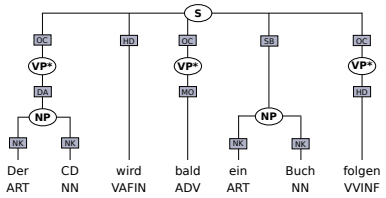
Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11), pages 126–134,
Paris, September 2012.



Crossing branches, original NeGra tree



Resolved crossing branches, NeGra distribution



Resolved crossing branches, Boyd (2007)

Figure 1: Crossing branches removal for NeGra. Note that the argument structure is changed as a result of the removal of the crossing branches.

discontinuities (Maier and Lichte, 2011). In LCFRS, a single non-terminal can span $k \geq 1$ continuous blocks of a string. A CFG is simply a special case of an LCFRS in which $k = 1$. k is called the *fan-out* of the non-terminal, and a corresponding constituent is said to have *block degree* k. It has been shown that probabilistic data-driven parsing on the basis of Probabilistic LCFRS (PLCFRS) is feasible and gives good results while preserving discontinuity information (Kallmeyer and Maier, 2010; Maier, 2010; van Cranenburgh et al., 2011; Evang and Kallmeyer, 2011; van Cranenburgh, 2012; Maier, 2012).

The major problem of PLCFRS parsing is its high computational complexity. A binarized PCFG can be parsed in $\mathcal{O}(n^3)$, parsing a binarized LCFRS takes $\mathcal{O}(n^{3k})$ (Seki et al., 1991), where k is the fan-out of the grammar (the maximal fan-out of any of its non-terminals). The parsers from the literature allow for an unbounded k. This leads to parsing times beyond practically acceptable values for sentences longer than 25 to 30 words.

In this paper, our goal is to show that by restricting the block degree, resp. the fan-out to two, (i) one can express almost all the information contained in the discontinuous treebank annotation of NeGra and the DPTB, and (ii) one can obtain a parser which is faster by an order of magnitude.

We proceed as follows. In section 2, we present definitions of PLCFRS, as well as of trees and the notion of block degree. In section 3, we describe how to bring the trees of both the DPTB and NeGra to block degree two. Unlike the transformations used for PCFG parsing, our transformations preserve the discontinuity information in almost all cases. Section 4 introduces PLCFRS as a formalism for data-driven parsing. In section 5, we present a data-driven parser for binary PLCFRS of fan-out two which uses an efficient case-by-case strategy, together with a new outside estimate for A* parsing. Section 6 contains experiments on the transformed NeGra as well as on the Discontinuous Penn Treebank. We use both the new parser and `rparse`, the parser used in our previous work (Kallmeyer and Maier, 2010). Our experiments show that given equal conditions, we achieve an enormous speed-up while obtaining an output of a comparable richness. Finally, section 7 concludes the article.

2 Definitions

We notate LCFRS with the syntax of Simple Range Concatenation Grammars (SRCG) (Boullier, 1998), a formalism equivalent to LCFRS.

An LCFRS (Vijay-Shanker et al., 1987) is a tuple $G = (N, T, V, P, S)$ where a) N is a finite set of non-terminals with a function dim: $N \to \mathbb{N}$ determining 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 rewriting rules

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

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 a *rank* $m \geq 0$. For all $r \in P$, every variable X occurring in r occurs exactly once in the left-hand side (LHS) and exactly once in the right-hand side (RHS). The *rank* of G is the maximal rank of any of its rules, its *fan-out* is the maximal fan-out of any of its non-terminals. If G has rank u and fan-out v, then G is an (u, v)-LCFRS.

$A(ab, cd) \to \varepsilon$ ($\langle ab, cd \rangle$ in yield of A)

$A(aXb, cYd) \to A(X, Y)$ (if $\langle X, Y \rangle$ in yield of A, then also $\langle aXb, cYd \rangle$ in yield of A)

$S(XY) \to A(X, Y)$ (if $\langle X, Y \rangle$ in yield of A, then $\langle XY \rangle$ in yield of S)

$L = \{a^n b^n c^n d^n \mid n > 0\}$

Figure 2: Sample LCFRS

A rewriting rule describes how to compute the yield of the LHS non-terminal from the yields of the RHS non-terminals. The yield of S is the language of the grammar. See figure 2 for a sample LCFRS.

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 a LCFRS and $p : P \to [0..1]$ a function such that for all $A \in N$: $\Sigma_{A(\vec{x}) \to \vec{\Phi} \in P} p(A(\vec{x}) \to \vec{\Phi}) = 1$.

A *tree* over a sentence $w = w_1 \cdots w_n$, $n \geq 1$, is a labeled ordered directed graph $\mathcal{D} = (V, E, r)$ with V a set of nodes, $E : V \times V$ a set of edges and $r \in V$ a single dedicated root node, where every $v \in V \setminus \{r\}$ has exactly one incoming edge and r has no incoming edges. All $v_l \in V$ with no outgoing edges are called *leaves* or *terminals*, and V_l is the set of all leaves or terminals. The *labeling* of $\mathcal{D}$ is given by a function $\Lambda : V \to N \cup \{1, \ldots, n\}$, where N a set of non-terminal labels, for all $v_i \in V_l$, $1 \leq i \leq n$, $\Lambda(v_i) = i$, and for all $v \in V \setminus V_l$, $\Lambda(v) \in N$. The function π gives the *yield* of the node; more precisely, for all $v \in V$, $\pi(v) = \{i \in \Lambda(u) \mid u \in V \text{ is a leaf and there is a } \langle v, u \rangle \in E^*\}$. The *ordering* of $\mathcal{D}$ is given by the relation $\prec$ which is such that for all v_1, v_2, $v_1 \prec v_2$ iff $min(\pi(v_1)) \leq min(\pi(v_2))$.

The yield *blocks* of v are given by a partition of $\pi(v)$ into maximal continuous sequences of integers. The *block degree* of v is the number of blocks of v, its *gap degree* is its block degree minus one. A gap of v is a tuple (i, k) such that $i \in \pi(v)$, $k + 1 \in \pi(v)$ and $j \notin \pi(v)$ for $i + 1 \leq j \leq k$.

3 Treebanks with Block-Degree Two

3.1 Removing Spurious Gaps

In the DPTB as used by Evang and Kallmeyer (2011),[2] the maximal block degree is three. Motivated by the suspicion (Evang, p.c.) that the cases of block degree three are spurious, i.e., caused only by punctuation, we move all punctuation terminals to the least common ancestor of their resp. left and right non-punctuation terminal neighbors. This is essentially the algorithm of Levy (2005), pp. 163. It leaves us with only 11 sentences containing nodes with more than one (non-spurious) gap. For our experiments, we remove those sentences; an investigation of their properties is left for future work.

In the NeGra annotation, punctuation and a very small number of other elements such as parts of ungrammatical sentences are not included in the annotation, i.e., the corresponding nodes are attached at the root node. They cause a very high, linguistically meaningless block degree of 40. In order to avoid gaps which contain nothing but those elements, we attach them lower.[3] Since aside from (punctuation) terminals, non-terminals may be concerned, we extend Levy's strategy as follows. Let n be a node originally attached to the root node, furthermore let $n_{l_1}, \ldots, n_{l_k}$, $n_{r_1}, \ldots, n_{r_m}$, $k, m \geq 0$, be all left, resp. right siblings of n for which it holds that both $\mathcal{S}_l = \{min(\pi(n))\} \cup (\bigcup_{i=1}^{k} \pi(n_{l_i}))$ and $\mathcal{S}_r = \{max(\pi(n))\} \cup (\bigcup_{j=1}^{m} \pi(n_{r_j}))$ are continuous sequences of integers. We select as an attachment target the least common ancestor node of the terminals t_l, t_r with $\pi(t_l) = \{(min(\mathcal{S}_l) - 1)\}$ and $\pi(t_r) = \{(max(\mathcal{S}_r) + 1)\}$. If t_r or t_l do not exist, we do not move n. This algorithm improves over the strategy from Maier (2012), pp. 189, in the sense that the latter does not remove all spurious gaps. We call the new strategy T_1.

3.2 Block Degree Two for NeGra

For NeGra, we introduce a novel series of linguistically motivated transformations which ensures that all resulting trees have block degree two. The block degrees of the treebank after each transformation are listed in table 1.

Verbs There is no consensus about the analysis of German verb phrases (VPs) and auxiliaries in particular, cf. Bouma and van Noord (1998) for a discussion. In the interest of a small block degree of the trees in NeGra, we change the VP anno-

[2] Thanks to Kilian Evang for providing us with his original data.

[3] This is a necessary preprocessing step for PCFG parsing as well since those elements are equally unattached in the version of NeGra with resolved crossing branches.

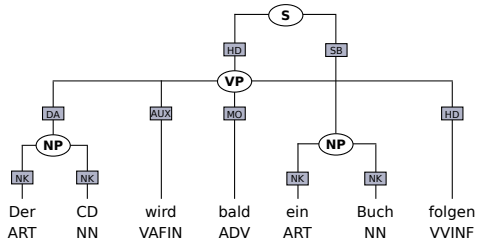


Figure 3: Verb transformation on original tree from fig. 1

Blocks	Orig.	T_1	T_2	T_3	T_4
1	7,704	14,927	10,898	10,944	10,944
2	5,275	4,988	9,333	9,361	9,658
3	3,585	679	370	297	-
4	1,917	8	1	-	-
5	998	-	-	-	-
≥ 6	1123	-	-	-	-

Table 1: Number of sentences in NeGra with a certain block degree before and after transformations

tation principles and group auxiliary verbs under the same VP as the corresponding full verb. We furthermore insert a VP for all finite verbs and their dependents except the subject (identified by the labels in the treebank), since in the original NeGra annotation, only non-finite verbs project to a VP. See figure 3 for an example.

The positions of the newly introduced VPs influences the attachment points for the punctuation attachment. As the second transformation T_2 we therefore first perform the verb transformation we just described, followed by T_1.

Parentheticals Parenthetical sentences such as (2) are annotated as embedding the enclosing sentence and therefore lead to an additional gap in the latter.

(2) …, so argumentierten die Richter, …
…, as argued the judges, …
"…, the judges argued, …"

We structurally identify them (not lexically) and attach them as low as necessary such that they do not create a gap. This is motivated by the annotation of TüBa-D/Z (Telljohann et al., 2012), another German treebank, where parenthetical sentences are left unattached. T_3 consists of T_2, followed by the parenthetical transformation.

Remainder Eventually, T_4 consists of T_3 followed by a transformation inspired by the standard crossing branches resolution for NeGra. We re-attach material to higher positions iff it causes a block degree higher than two. Thereby, we first consider sentential modifiers, then modifiers in general and only finally constituents of any sort. T_4 only treats a tiny fraction of all sentences; however, it does change structures for which a block degree higher than two can be linguistically justified, such as di-transitive adjectives and verbs in particular word order configurations. A more careful investigation of T_4 is left for future work.

4 PLCFRS for Data-Driven Parsing

LCFRSs can be extracted directly from treebanks with a direct annotation of discontinuities (Maier and Søgaard, 2008). The difference between treebank PLCFRS and PCFG extraction is, intuitively, that in PLCFRS variables are used to describe the blocks which are dominated by a non-terminal. In other words, an argument boundary in a production corresponds to a block boundary of the corresponding non-terminal in the tree, and the fan-out of an extracted rule is equal to the block degree of the treebank non-terminal corresponding to the rule's LHS non-terminal. Consider again the original tree from figure 1. From the discontinuous VP *Der CD …bald …folgen* we extract the rule $VP(X_1, X_2, X_3) \rightarrow NP(X_1)\,ADV(X_2)\,VVINF(X_3)$. The LHS non-terminal has fan-out three due to the fact that the VP has block degree three.

When applied to a treebank of block degree two, the extraction algorithm yields grammars of fan-out two. In order to obtain a $(2,2)$-PLCFRS, i.e., for rank reduction, we use the *optimal binarization* algorithm of Kallmeyer (2010), p. 150, which yields a minimal fan-out, resp. number of variables per binarized rule. As in PCFG parsing, we use markovization (Kallmeyer and Maier, 2010). We use standard Maximum Likelihood estimation. See section 6 for further experimental details.

5 A CYK Parser for $(2,2)$-PLCFRS

5.1 The Parser

Just as Kallmeyer and Maier (2010), we use a probabilistic CYK parser (Seki et al., 1991). The general CYK deduction system is shown in figure 4. Its items have the form $[A, \vec{\rho}]$, with $A \in N$ and

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

Unary: $\frac{in : [B, \vec{\rho}]}{in + |\log(p)| : [A, \vec{\rho}]}$ $p : A(\vec{\rho}) \to B(\vec{\rho}) \in P$

Binary: $\frac{in_B : [B, \vec{\rho_B}], in_C : [C, \vec{\rho_C}]}{in_B + in_C + |\log(p)| : [A, \vec{\rho_A}]}$

where $p : A(\vec{\rho_A}) \to B(\vec{\rho_B})C(\vec{\rho_C})$ is an instantiated rule.

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

Figure 4: Weighted CYK deduction system for LCFRS

ID	Type	G30T	E30
1	A(X) → B(X)	49	235
2	A(X,Y) → B(X,Y)	1	4
3	A(XY) → B(X) C(Y)	14,430	11,777
4	A(X,Y) → B(X) C(Y)	1,644	312
5	A(XYZ) → B(X,Z) C(Y)	621	205
6	A(X,YZ) → B(X,Y) C(Z)	100	45
7	A(X,YZ) → B(X,Z) C(Y)	142	94
8	A(XY,Z) → B(X,Z) C(Y)	172	10
9	A(XY,Z) → B(X) C(Y,Z)	582	108
10	A(XY,ZU) → B(X,Z) C(Y,U)	7	0
11	A(XY,ZU) → B(X,U) C(Y,Z)	0	0
12	A(X,YZU) → B(X,Z) C(Y,U)	12	3
13	A(XYZ,U) → B(X,Z) C(Y,U)	12	2
14	A(XYZU) → B(X,Z) C(Y,U)	13	6

Figure 5: LCFRS rule types and numbers of occurrence in binarized grammars (cf. section 6)

$\vec{\rho}$ a vector of ranges characterizing all components of the span of A. We specify a simpler, specialized deduction system which takes advantage of the fact that due to our maximum fan-out of two, we can rely on only encountering rules of certain forms. The second column of figure 5 schematically displays all 14 different rule types the parser must handle.

In the specialized deduction system, *unary items* now take the form $[A, i, j]$ and *binary items* take the form $[A, i, j, k, l]$, where $A \in N$ and i, j, resp. k, l are spans dominated by A with $0 \leq i < j < k < l \leq n$. The goal item is $[S, 0, n]$. We replace the old Unary and Binary deduction rules in figure 4 with 14 new rules, one per production type. Figure 6 shows the new scan rule and the complete rules for type 1, type 6 and type 10, which should make the basic idea clear. Note that there is no need to refer to instantiations anymore. Our case-by-case strategy is similar to the one employed by Kato et al. (2006).

As in our previous work, we specify the set

Scan: $\frac{}{0 : [A, i, i+1]}$ A POS tag of w_{i+1}

Complete1: $\frac{in : [B, i, j]}{in + |\log(p)| : [A, i, j]}$

where $p : A(X) \to B(X) \in P$.

Complete6: $\frac{in_B : [B, i, j, k, l], in_C : [C, l, u]}{in_B + in_C + |\log(p)| : [A, i, j, k, u]}$

where $p : A(X, YZ) \to B(X, Y)C(Z) \in P$.

Complete10: $\frac{in_B : [B, i, x, k, y], in_C : [C, x, j, y, l]}{in_B + in_C + |\log(p)| : [A, i, j, k, l]}$

where $p : A(XY, ZU) \to B(X, Z)C(Y, U) \in P$

Figure 6: Weighted CYK deduction rules for 2-LCFRS

of parse items using the algorithm of weighted deductive parsing (WDP) (Nederhof, 2003). In WDP, one maintains a priority queue of items, sorted by the resp. Viterbi inside scores. The topmost item is always processed first. WDP guarantees optimality, i.e., that the best parse is found.

5.2 A Novel Outside Estimate

One can speed up parsing by adding to the Viterbi inside score of an item an estimate of its Viterbi outside score, in other words, an estimate of the cost of completion of the item to a complete parse. This has proven to be successful for both PCFG (Klein and Manning, 2003) and PLCFRS (Kallmeyer and Maier, 2010). As outside estimate, one uses the outside probability of a summary of the item, i.e., of an equivalence class of parse items. The difficulty for PLCFRS is to choose the summary such that optimality is maintained through the two estimate properties *admissibility* and *monotonicity* (Klein and Manning, 2003).

Here, we present the novel *LN estimate*, which is based on a summary that records only the sum of the span lengths and the length of the entire sentence. It is the first practically computable estimate which allows for maintaining optimality.

The estimate is computed offline up to a certain maximal sentence length len_{max}. We specify the estimate computation with the deduction system in figure 7.[4] The items have the form $[X, len, slen]$ with $X \in N$, $dim(X) \leq len \leq slen$. The value $in(X, l)$ for a non-terminal X and a length l, $0 \leq l \leq len_{max}$ is an estimate of

[4] A simpler deduction system for the estimate computation for $(2, 2)$-LCFRS would be possible as well, along the lines of the simplification of the CYK parser.

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

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

$$\text{Binary-right}: \frac{w{:}[X,l_X,slen]}{w+in(A,l_X-l_B)+|\log(p)|{:}[B,l_B,slen]}$$

$$\text{Binary-left}: \frac{w{:}[X,l_X,slen]}{w+in(B,l_X-l_A)+|\log(p)|{:}[A,l_A,slen]}$$
where, for both rules,
$p : X(\vec{\alpha}) \rightarrow A(\vec{\alpha_A})B(\vec{\alpha_B}) \in P$.

Figure 7: LN estimate (span and sentence length)

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

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

$$\text{Binary}: \frac{in_B{:}[B,l_B],in_C{:}[C,l_C]}{in_B+in_C+|\log(p)|{:}[A,l_B+l_C]}$$
where either $p : A(\vec{\alpha_A}) \rightarrow B(\vec{\alpha_B})C(\vec{\alpha_C}) \in P$ or
$p : A(\vec{\alpha_A}) \rightarrow C(\vec{\alpha_C})B(\vec{\alpha_B}) \in P$.

Figure 8: Inside estimate with total span length

the inside score of an X category with a span of length l. Its computation is specified in figure 8.

The outside 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 minimal weight of $[C, len, n]$.

We will show in the following that the LN estimate maintains optimal search by being both admissible and monotonic. Since the weight of the outside estimate for an item is always lower or equal to the actual outside probability, given the input, the weight of an item in the agenda is always lower or equal to the log of the actual product of inside and outside probability of the constituent represented by the item. Therefore, the LN estimate is admissible. In order to prove that the estimate is also monotonic, we look at the CYK deduction rules when being augmented with the estimate. Only *Unary* and *Binary* are relevant since *Scan* does not have antecedent items. The two rules are now as follows:

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

$$\text{Binary}: \frac{in_B+out_B{:}[B,\vec{\rho_B}],in_C+out_C{:}[C,\vec{\rho_C}]}{in_B+in_C+|\log(p)|+out_A{:}[A,\vec{\rho_A}]}$$
where $p : A(\vec{\rho_A}) \rightarrow B(\vec{\rho_B})C(\vec{\rho_C})$ is an instantiated rule. (Here, out_A, out_B and out_C are the respective outside estimates of $[A, \vec{\rho_A}]$, $[B, \vec{\rho_B}]$ and $[C, \vec{\rho_C}]$.)

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 $w \le w'$.

We start with *Unary*. To show: $in_B + out_B \le in_B + |\log(p)| + out_A$. Because of the *Unary* rule for computing the outside estimate and because of the unary production, 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 most $out_A + |\log(p)|$, i.e., $out_B \le |\log(p)| + out_A$. □

Now we 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: $in_C + out_C \le in_B + in_C + |\log(p)| + out_A$. 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 most $out_A + in(B, l_B) + |\log(p)|$. Furthermore, $in(B, l_B) \le in_B$. Consequently $out_C \le in_B + |\log(p)| + out_A$. □

6 Experiments

We have implemented the parser within the API of *rparse* in order to provide equal conditions. The new parser will be made available under GNU GPL.[5] For all experiments, we have used the newest Oracle Java 7, running on Debian Linux on a series of Intel Xeon X5690 nodes at 3.46GHz.

6.1 Data and Experimental Setup

We perform experiments with both the English DPTB and the German NeGra. The names of the data sets will have the prefixes *E* (for the DPTB)

[5] See `http://www.phil.hhu.de/rparse` for more information.

and *G* (for NeGra). We create two versions of NeGra in which we limit the sentence lengths to 30 and 40 words respectively and investigate the treebank after T_4 (data set name suffix *T*) (only for 30 words) and after T_1 (data set name suffix *O*) (for 30 and 40 words). The names of the data sets are consequently: G30O (for the 30-word data set after T_1) and G30T, resp. G40T (for the 30- and 40-word data sets after T_4). As for the DPTB, we create one data set E30 with a sentence length limit of 30. In E30, we reattach punctuation tokens as described in section 3.1. For training, resp. testing we use the first 90%, resp. the last 10% of each data set. The parser is provided with gold POS tags.

We extract PLCFRSs from our data sets as described before and binarize them using the optimal binarization algorithm from Kallmeyer (2010). For E30 we cannot resort to deterministic left-to-right binarization as done by Evang and Kallmeyer, since it results in a binarized grammar of fan-out three. Note that in general, given an unbinarized LCFRS production with a fan-out of two, finding a binarization which does not increase the fan-out cannot be guaranteed if its RHS has a length > 3 (Gómez-Rodríguez et al., 2010; Rambow and Satta, 1999). However, with the optimal algorithm, we have not observed an increased fan-out in practice, neither for NeGra, nor for the DPTB. Figure 5 shows the occurrence counts of the 14 different production types in the binarized grammars of G30T and E30. For the choice of the remaining parsing parameters, we exploit the results of Maier (2012): We do not use unary rules during binarization and markovize the binarized grammars with $v = 1$, $h = 2$.

6.2 Parsing Speed

We first investigate the speed of the new parser on both NeGra and the DPTB.

NeGra The upper graph in figure 9 shows the average parsing times of both parsers on G40T. The speed-up provided by the case-by-case strategy of the new parser is enormous. The average parsing time for a sentence of length 40 (a common upper length limit in PCFG parsing literature, see, e.g., Klein and Manning (2003)) drops from several hours with `rparse` to slightly under 3 minutes with the new parser. Note that the parsing complexity is not changed. The speed gain

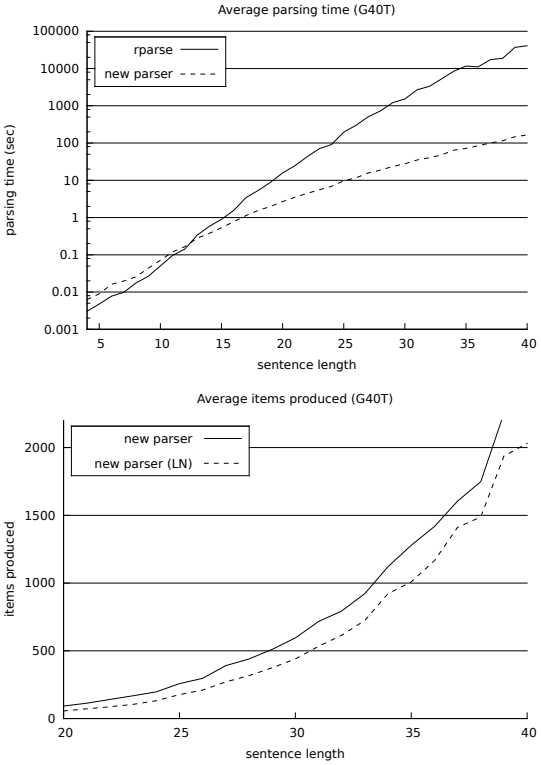


Figure 9: Average parsing times and items for G40T

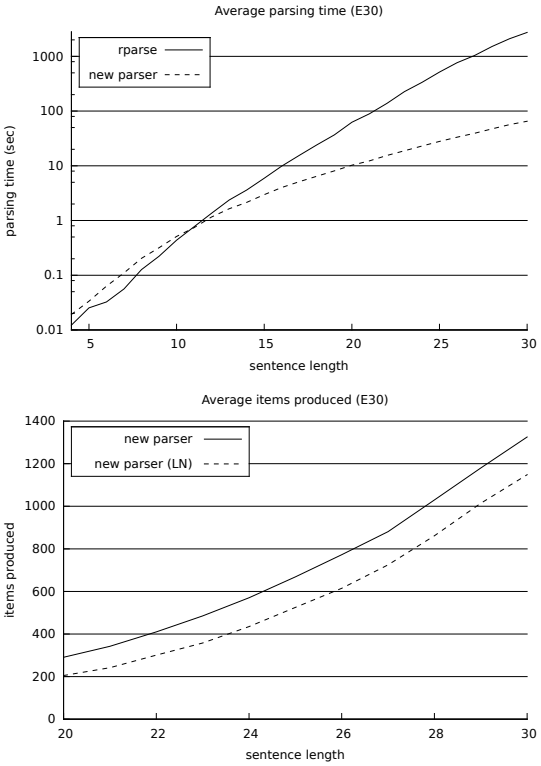


Figure 10: Average parsing times and items for E30

	Precision	Recall	F_1
G30O	74.6	74.5	74.5
G30T	71.8	71.7	71.8
G30O-S	73.6	73.8	73.7
G30T-V	73.5	73.6	73.5

Table 2: Parsing results for NeGra

can be attributed to the fact that it is much cheaper to perform the simple integer comparisons of the specialized *Complete* rules (fig. 6) than to provide a comparison operation for range vectors of an arbitrary length (Maier, 2012, p. 176). This becomes more clear when regarding the pseudo-code formulation of the similar case-by-case strategy of Kato et al. (2006).

As for the LN estimate, we can observe that it effectively reduces the number of items which are produced (cf. the lower graph in fig. 9). However, it has less effect than the estimates presented in previous work (Kallmeyer and Maier, 2010). This indicates that the context summary consisting of the sum of the span lengths and the total sentence length provides too few information. For $(2, 2)$-LCFRS, unlike for full LCFRS, the full SX estimate from Kallmeyer and Maier should be computable and should deliver better results. We postpone this to future work.

DPTB The upper graph in figure 10 shows the average parsing times for both parsers on E30. We can see that the speed gain with the new parser is similar to the one we obtain on NeGra. The behavior of the LN estimate is also similar to its behavior in the NeGra experiments (cf. the lower graph in fig. 10).

6.3 Output Quality

For the qualitative evaluation of the parser output, we use the extended `evalb` measure for PLCFRS (Maier, 2010). We report labeled precision, recall and F_1.

NeGra In order to investigate how the transformed treebank behaves compared to the unmodified treebank, we run `rparse` on G30O and the new parser on G30T. Intuitively, one might expect that the less flat annotation of the transformed treebank leads to better results (Rehbein and van Genabith, 2007), however, as can be seen in table 2, the results on G30T are worse. We can identify two major reasons for this: The status of subjects and different types of verb phrases.

Subjects can be identified structurally in the transformed treebank, because they are attached below S while other arguments are part of the newly introduced VPs. In the original treebank, when disregarding grammatical functions (such as we do), subject NPs are indistinguishable from other NPs. In other words, with the transformed treebank, the parser must cope with the additional tasks of identifying subjects. We therefore produce a minimally modified version of G30O, G30O-S, in which subjects can be identified by node labels. In the original annotation, the edge label SB designates subjects. We rename all NPs with an SB edge to NP-SB. Subjects which consists only of a single word are attached directly to the sentence in the original annotation, we project them to a new single NP-SB node instead. The results get about 0.8 points worse (cf. tab. 2), reflecting the difficulty of the task.

Verb phrases also have a different status in the transformed treebank. While per definition in the original annotation, the VP label only designates non-finite VPs, in the transformed treebank, we have both finite and non-finite VPs. We therefore produce a modified version of G30T, G30T-V, in which we change the label of a VP to VPFIN if it has a finite lexical head. Similar linguistically motivated splits have successfully been used before (Maier, 2010). It turns out that the results for G30T-V and G30O-S lie very close together (again cf. tab. 2).

DPTB For the sake of completeness we report the results for the DPTB as well. On E30, we obtain LP 76.15, LR 70.94, and therefore a LF_1 of 73.45. Our parameter settings have not been tried before on the DPTB (Evang, 2011; Evang and Kallmeyer, 2011), therefore there are no previous result to compare to.

7 Conclusion

The goal of this paper on data-driven PLCFRS parsing was to show that by restricting the block degree of trees used for grammar extraction, resp. the fan-out of the resulting grammars to two, (i) one can express almost all the information contained in the discontinuous treebank annotation of NeGra and the DPTB, and (ii) one obtains a parser which is much faster than a parser for general

LCFRS on the same data. The first contribution of this paper is a series of treebank transformations for NeGra and the DPTB which produces trees of a block degree of at most two. Unlike transformations for PCFG parsing, our transformations almost completely preserve the annotation information on discontinuities. The second contribution is an efficient data-driven parser for $(2, 2)$-PLCFRS, to be extracted from the converted treebanks. The evaluation of experiments with this parser on both NeGra and the Penn Treebank shows that an enormous speed-up has been achieved in comparison to earlier PLCFRS parsers, all while obtaining an output of comparable richness.

References

Pierre Boullier. 1998. Proposal for a Natural Language Processing syntactic backbone. Technical Report 3342, INRIA.

Gosse Bouma and Gertjan van Noord. 1998. Word Order Constraints on Verb Clusters in German and Dutch. In Erhard Hinrichs, Tsuneko Nakazawa, and Andreas Kathol, editors, *Complex predicates in Nonderivational Syntax*. Academic Press.

Adriane Boyd. 2007. Discontinuity revisited: An improved conversion to context-free representations. In *Proceedings of The Linguistic Annotation Workshop*.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank. In *Proceedings of TLT*.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of English discontinuous constituents. In *Proceedings of IWPT*.

Kilian Evang. 2011. Parsing discontinuous constituents in English. Master's thesis, University of Tübingen.

Carlos Gómez-Rodríguez, Marco Kuhlmann, and Giorgio Satta. 2010. Efficient parsing of well-nested Linear Context-Free Rewriting Systems. In *Proceedings of HLT-NAACL*.

Geoffrey Huck and Almerindo Ojeda, editors. 1987. *Discontinuous constituency*. Academic Press.

Laura Kallmeyer and Wolfgang Maier. 2010. Data-driven parsing with Probabilistic Linear Context-Free Rewriting Systems. In *Proceedings of COLING*.

Laura Kallmeyer. 2010. *Parsing beyond Context-Free Grammar*. Springer.

Yuki Kato, Hiroyuki Seki, and Tadao Kasami. 2006. Stochastic multiple Context-Free Grammar for RNA pseudoknot modeling. In *Proceedings of TAG+8*.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact viterbi parse selection. In *Proceedings of NAACL*.

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

Wolfgang Maier and Timm Lichte. 2011. Characterizing discontinuity in constituent treebanks. In *FG 2009, Revised Selected Papers*, volume 5591 of *LNAI*. Springer.

Wolfgang Maier and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In Philippe de Groote, editor, *Proceedings of Formal Grammar*. CSLI.

Wolfgang Maier. 2010. Direct parsing of discontinuous constituents in German. In *Proceedings of SPMRL*.

Wolfgang Maier. 2012. *Parsing Discontinuous Structures*. Ph.D. thesis, University of Tübingen.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Mark-Jan Nederhof. 2003. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):1–9.

Owen Rambow and Giorgio Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223(1-2):87–120.

Ines Rehbein and Josef van Genabith. 2007. Evaluating evaluation measures. In *Proceedings of NODALIDA-2007*.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On Multiple Context-Free Grammars. *Theoretical Computer Science*, 88(2):191–229.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of ANLP*, pages 88–95.

Heike Telljohann, Erhard W. Hinrichs, Sandra Kübler, Heike Zinsmeister, and Kathrin Beck. 2012. Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z). Technical report, University of Tübingen.

Andreas van Cranenburgh, Remko Scha, and Federico Sangati. 2011. Discontinuous data-oriented parsing: A mildly context-sensitive all-fragments grammar. In *Proceedings of SPMRL*.

Andreas van Cranenburgh. 2012. Efficient parsing with linear context-free rewriting systems. In *Proceedings of EACL*.

K. Vijay-Shanker, David Weir, and Aravind K. Joshi. 1987. Characterising structural descriptions used by various formalisms. In *Proceedings of ACL*.