An Optimal-Time Binarization Algorithm for Linear Context-Free Rewriting Systems with Fan-Out Two

Carlos Gómez-Rodríguez
Departamento de Computación
Universidade da Coruña, Spain
cgomezr@udc.es

Giorgio Satta
Department of Information Engineering
University of Padua, Italy
satta@dei.unipd.it

Abstract

Linear context-free rewriting systems (LCFRSs) are grammar formalisms with the capability of modeling discontinuous constituents. Many applications use LCFRSs where the *fan-out* (a measure of the discontinuity of phrases) is not allowed to be greater than 2. We present an efficient algorithm for transforming LCFRS with fan-out at most 2 into a binary form, whenever this is possible. This results in asymptotical run-time improvement for known parsing algorithms for this class.

1 Introduction

Since its early years, the computational linguistics field has devoted much effort to the development of formal systems for modeling the syntax of natural language. There has been a considerable interest in rewriting systems that enlarge the generative power of context-free grammars, still remaining far below the power of the class of context-sensitive grammars; see (Joshi et al., 1991) for discussion. Following this line, (Vijay-Shanker et al., 1987) have introduced a formalism called linear context-free rewriting systems (LCFRSs) that has received much attention in later years by the community.

LCFRSs allow the derivation of tuples of strings,[1] i.e., discontinuous phrases, that turn out to be very useful in modeling languages with relatively free word order. This feature has recently been used for mapping non-projective dependency grammars into discontinuous phrase structures (Kuhlmann and Satta, 2009). Furthermore, LCFRSs also implement so-called synchronous rewriting, up to some bounded degree, and have recently been exploited, in some syntactic variant, in syntax-based machine translation (Chiang, 2005; Melamed, 2003) as well as in the modeling of syntax-semantic interface (Nesson and Shieber, 2006).

The maximum number f of tuple components that can be generated by an LCFRS G is called the **fan-out** of G, and the maximum number r of nonterminals in the right-hand side of a production is called the **rank** of G. As an example, context-free grammars are LCFRSs with $f = 1$ and r given by the maximum length of a production right-hand side. Tree adjoining grammars (Joshi and Levy, 1977), or TAG for short, can be viewed as a special kind of LCFRS with $f = 2$, since each elementary tree generates two strings, and r given by the maximum number of adjunction sites in an elementary tree.

Several parsing algorithms for LCFRS or equivalent formalisms are found in the literature; see for instance (Seki et al., 1991; Boullier, 2004; Burden and Ljunglöf, 2005). All of these algorithms work in time $\mathcal{O}(|G| \cdot |w|^{f \cdot (r+1)})$. Parsing time is then exponential in the input grammar size, since $|G|$ depends on both f and r. In the development of efficient algorithms for parsing based on LCFRS the crucial goal is therefore to optimize the term $f \cdot (r + 1)$.

In practical natural language processing applications the fan-out of the grammar is typically bounded by some small number. As an example, in the case of discontinuous parsing discussed above, we have $f = 2$ for most practical cases. On the contrary, LCFRS productions with a relatively large number of nonterminals are usually observed in real data. The reduction of the rank of a LCFRS, called **binarization**, is a process very similar to the reduction of a context-free grammar into Chomsky normal form. While in the special case of CFG and TAG this can always be achieved,

[1] In its more general definition, an LCFRS provides a framework where abstract structures can be generated, as for instance trees and graphs. Throughout this paper we focus on so-called string-based LCFRSs, where rewriting is defined over strings only.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 985–993,
Suntec, Singapore, 2-7 August 2009.

binarization of an LCFRS requires, in the general case, an increase in the fan-out of the grammar much larger than the achieved reduction in the rank. Worst cases and some lower bounds have been discussed in (Rambow and Satta, 1999; Satta, 1998).

Nonetheless, in many cases of interest binarization of an LCFRS can be carried out without any extra increase in the fan-out. As an example, in the case where $f = 2$, binarization of a LCFRS would result in parsing time of $\mathcal{O}(|G| \cdot |w|^6)$. With the motivation of parsing efficiency, much research has been recently devoted to the design of efficient algorithms for rank reduction, in cases in which this can be carried out at no extra increase in the fan-out. (Gómez-Rodríguez et al., 2009) reports a general binarization algorithm for LCFRS. In the case where $f = 2$, this algorithm works in time $\mathcal{O}(|p|^7)$, where p is the input production. A more efficient algorithm is presented in (Kuhlmann and Satta, 2009), working in time $\mathcal{O}(|p|)$ in case of $f = 2$. However, this algorithm works for a restricted typology of productions, and does not cover all cases in which some binarization is possible. Other linear time algorithms for rank reduction are found in the literature (Zhang et al., 2008), but they are restricted to the case of synchronous context-free grammars, a strict subclass of the LCFRS with $f = 2$.

In this paper we focus our attention on LCFRS with a fan-out of two. We improve upon all of the above mentioned results, by providing an algorithm that computes a binarization of an LCFRS production in all cases in which this is possible and works in time $\mathcal{O}(|p|)$. This is an optimal result in terms of time complexity, since $\Theta(|p|)$ is also the size of any output binarization of an LCFRS production.

2 Linear context-free rewriting systems

We briefly summarize here the terminology and notation that we adopt for LCFRS; for detailed definitions, see (Vijay-Shanker et al., 1987). We denote the set of non-negative integers by $\mathbb{N}$. For $i, j \in \mathbb{N}$, the **interval** $\{k \mid i \leq k \leq j\}$ is denoted by $[i, j]$. We write $[i]$ as a shorthand for $[1, i]$. For an alphabet V, we write V^* for the set of all (finite) strings over V.

As already mentioned in Section 1, linear context-free rewriting systems generate tuples of strings over some finite alphabet. This is done by associating each production p of a grammar with a function g that rearranges the string components in the tuples generated by the nonterminals in p's right-hand side, possibly adding some alphabet symbols. Let V be some finite alphabet. For natural numbers $r \geq 0$ and $f, f_1, \ldots, f_r \geq 1$, consider a function $g : (V^*)^{f_1} \times \cdots \times (V^*)^{f_r} \rightarrow (V^*)^f$ defined by an equation of the form

$$g(\langle x_{1,1}, \ldots, x_{1,f_1} \rangle, \ldots, \langle x_{r,1}, \ldots, x_{r,f_r} \rangle) = \vec{\alpha},$$

where $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_f \rangle$ is an f-tuple of strings over g's argument variables and symbols in V. We say that g is **linear, non-erasing** if $\vec{\alpha}$ contains *exactly one* occurrence of each argument variable. We call r and f the **rank** and the **fan-out** of g, respectively, and write $r(g)$ and $f(g)$ to denote these quantities.

A **linear context-free rewriting system** (LCFRS) is a tuple $G = (V_N, V_T, P, S)$, where V_N and V_T are finite, disjoint alphabets of nonterminal and terminal symbols, respectively. Each $A \in V_N$ is associated with a value $f(A)$, called its **fan-out**. The nonterminal S is the start symbol, with $f(S) = 1$. Finally, P is a set of productions of the form

$$p : A \rightarrow g(A_1, A_2, \ldots, A_{r(g)}),$$

where $A, A_1, \ldots, A_{r(g)} \in V_N$, and $g : (V_T^*)^{f(A_1)} \times \cdots \times (V_T^*)^{f(A_{r(g)})} \rightarrow (V_T^*)^{f(A)}$ is a linear, non-erasing function.

A production p of G can be used to transform a sequence of $r(g)$ string tuples generated by the nonterminals $A_1, \ldots, A_{r(g)}$ into a tuple of $f(A)$ strings generated by A. The values $r(g)$ and $f(g)$ are called the **rank** and **fan-out** of p, respectively, written $r(p)$ and $f(p)$. The rank and fan-out of G, written $r(G)$ and $f(G)$, respectively, are the maximum rank and fan-out among all of G's productions. Given that $f(S) = 1$, S generates a set of strings, defining the language of G.

Example 1 Consider the LCFRS G defined by the productions

$$p_1 : S \rightarrow g_1(A), \quad g_1(\langle x_{1,1}, x_{1,2} \rangle) = \langle x_{1,1} x_{1,2} \rangle$$
$$p_2 : A \rightarrow g_2(A), \quad g_2(\langle x_{1,1}, x_{1,2} \rangle) = \langle a x_{1,1} b, c x_{1,2} d \rangle$$
$$p_3 : A \rightarrow g_3(), \quad g_3() = \langle \varepsilon, \varepsilon \rangle$$

We have $f(S) = 1$, $f(A) = f(G) = 2$, $r(p_3) = 0$ and $r(p_1) = r(p_2) = r(G) = 1$. G generates the string language $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$. For instance, the string $a^3 b^3 c^3 d^3$ is generated by means

of the following bottom-up process. First, the tuple $\langle \varepsilon, \varepsilon \rangle$ is generated by A through p_3. We then iterate three times the application of p_2 to $\langle \varepsilon, \varepsilon \rangle$, resulting in the tuple $\langle a^3b^3, c^3d^3 \rangle$. Finally, the tuple (string) $\langle a^3b^3c^3d^3 \rangle$ is generated by S through application of p_1. □

3 Position sets and binarizations

Throughout this section we assume an LCFRS production $p : A \to g(A_1, \ldots, A_r)$ with g defined through a tuple $\vec{\alpha}$ as in section 2. We also assume that the fan-out of A and the fan-out of each A_i are all bounded by two.

3.1 Production representation

We introduce here a specialized representation for p. Let \$ be a fresh symbol that does not occur in p. We define the **characteristic string** of p as the string

$$\sigma_N(p) = \alpha'_1 \$ \alpha'_2 \$ \cdots \$ \alpha'_{f(A)},$$

where each α'_j is obtained from α_j by removing all the occurrences of symbols in V_T. Consider now some occurrence A_i of a nonterminal symbol in the right-hand side of p. We define the **position set** of A_i, written X_{A_i}, as the set of all non-negative integers $j \in [|\sigma_N(p)|]$ such that the j-th symbol in $\sigma_N(p)$ is a variable of the form $x_{i,h}$ for some h.

Example 2 Let $p : A \to g(A_1, A_2, A_3)$, where $g(\langle x_{1,1}, x_{1,2} \rangle, \langle x_{2,1} \rangle, \langle x_{3,1}, x_{3,2} \rangle) = \vec{\alpha}$ with

$$\vec{\alpha} = \langle x_{1,1} a x_{2,1} x_{1,2},\ x_{3,1} b x_{3,2} \rangle .$$

We have $\sigma_N(p) = x_{1,1}x_{2,1}x_{1,2}\$x_{3,1}x_{3,2}$, $X_{A_1} = \{1, 3\}$, $X_{A_2} = \{2\}$ and $X_{A_3} = \{5, 6\}$. □

Each position set $X \subseteq [|\sigma_N(p)|]$ can be represented by means of non-negative integers $i_1 < i_2 < \cdots < i_{2k}$ satisfying

$$X = \bigcup_{j=1}^{k} [i_{2j-1} + 1, i_{2j}].$$

In other words, we are decomposing X into the union of k intervals, with k as small as possible. It is easy to see that this decomposition is always unique. We call set $E = \{i_1, i_2, \ldots, i_{2k}\}$ the **endpoint set** associated with X, and we call k the **fan-out** of X, written $f(X)$. Throughout this paper, we will represent p as the collection of all the position sets associated with the occurrences of nonterminals in its right-hand side.

Let X_1 and X_2 be two disjoint position sets (i.e., $X_1 \cap X_2 = \emptyset$), with $f(X_1) = k_1$ and $f(X_2) = k_2$ and with associated endpoint sets E_1 and E_2, respectively. We define the **merge** of X_1 and X_2 as the set $X_1 \cup X_2$. We extend the position set and end-point set terminology to these merge sets as well. It is easy to check that the endpoint set associated to position set $X_1 \cup X_2$ is $(E_1 \cup E_2) \setminus (E_1 \cap E_2)$. We say that X_1 and X_2 are **2-combinable** if $f(X_1 \cup X_2) \leq 2$. We also say that X_1 and X_2 are **adjacent**, written $X_1 \leftrightarrow X_2$, if $f(X_1 \cup X_2) \leq \max(k_1, k_2)$. It is not difficult to see that $X_1 \leftrightarrow X_2$ if and only if X_1 and X_2 are disjoint and $|E_1 \cap E_2| \geq \min(k_1, k_2)$. Note also that $X_1 \leftrightarrow X_2$ always implies that X_1 and X_2 are 2-combinable (but not the other way around).

Let $\mathcal{X}$ be a collection of mutually disjoint position sets. A **reduction** of $\mathcal{X}$ is the process of merging two position sets $X_1, X_2 \in \mathcal{X}$, resulting in a new collection $\mathcal{X}' = (\mathcal{X} \setminus \{X_1, X_2\}) \cup \{X_1 \cup X_2\}$. The reduction is **2-feasible** if X_1 and X_2 are 2-combinable. A **binarization** of $\mathcal{X}$ is a sequence of reductions resulting in a new collection with two or fewer position sets. The binarization is 2-feasible if all of the involved reductions are 2-feasible. Finally, we say that $\mathcal{X}$ is 2-feasible if there exists at least one 2-feasible binarization for $\mathcal{X}$.

As an important remark, we observe that when a collection $\mathcal{X}$ represents the position sets of all the nonterminals in the right-hand side of a production p with $r(p) > 2$, then a 2-feasible reduction merging $X_{A_i}, X_{A_j} \in \mathcal{X}$ can be interpreted as follows. We replace p by means of a new production p' obtained from p by substituting A_i and A_j with a fresh nonterminal symbol B, so that $r(p') = r(p) - 1$. Furthermore, we create a new production p'' with A_i and A_j in its right-hand side, such that $f(p'') = f(B) \leq 2$ and $r(p'') = 2$. Productions p' and p'' together are equivalent to p, but we have now achieved a local reduction in rank of one unit.

Example 3 Let p be defined as in example 2 and let $\mathcal{X} = \{X_{A_1}, X_{A_2}, X_{A_3}\}$. We have that X_{A_1} and X_{A_2} are 2-combinable, and their merge is the new position set $X = X_{A_1} \cup X_{A_2} = \{1, 2, 3\}$. This merge corresponds to a 2-feasible reduction of $\mathcal{X}$ resulting in $\mathcal{X}' = \{X, X_{A_3}\}$. Such a reduction corresponds to the construction of a new production $p' : A \to g'(B, A_3)$ with

$$g'(\langle x_{1,1} \rangle, \langle x_{3,1}, x_{3,2} \rangle) = \langle x_{1,1}, x_{3,1} b x_{3,2} \rangle ;$$

and a new production $p'' : B \rightarrow g''(A_1, A_2)$ with

$$g''(\langle x_{1,1}, x_{1,2}\rangle, \langle x_{2,1}\rangle) = \langle x_{1,1} a x_{2,1} x_{1,2}\rangle \,. \quad \square$$

It is easy to see that $\mathcal{X}$ is 2-feasible if and only if there exists a binarization of p that does not increase its fan-out.

Example 4 It has been shown in (Rambow and Satta, 1999) that binarization of an LCFRS G with $f(G) = 2$ and $r(G) = 3$ is always possible without increasing the fan-out, and that if $r(G) \geq 4$ then this is no longer true. Consider the LCFRS production $p : A \rightarrow g(A_1, A_2, A_3, A_4)$, with $g(\langle x_{1,1}, x_{1,2}\rangle, \langle x_{2,1}, x_{2,2}\rangle, \langle x_{3,1}, x_{3,2}\rangle, \langle x_{4,1}, x_{4,2}\rangle) = \vec{\alpha}$, $\vec{\alpha} = \langle x_{1,1} x_{2,1} x_{3,1} x_{4,1},\ x_{2,2} x_{4,2} x_{1,2} x_{3,2}\rangle$. It is not difficult to see that replacing any set of two or three nonterminals in p's right-hand side forces the creation of a fresh nonterminal of fan-out larger than two. $\square$

3.2 Greedy decision theorem

The binarization algorithm presented in this paper proceeds by representing each LCFRS production p as a collection of disjoint position sets, and then finding a 2-feasible binarization of p. This binarization is computed deterministically, by an iterative process that greedily chooses merges corresponding to pairs of *adjacent* position sets.

The key idea behind the algorithm is based on a theorem that guarantees that any merge of adjacent sets preserves the property of 2-feasibility:

Theorem 1 *Let $\mathcal{X}$ be a 2-feasible collection of position sets. The reduction of $\mathcal{X}$ by merging any two adjacent position sets $D_1, D_2 \in \mathcal{X}$ results in a new collection $\mathcal{X}'$ which is 2-feasible.*

To prove Theorem 1 we consider that, since $\mathcal{X}$ is 2-feasible, there must exist at least one 2-feasible binarization for $\mathcal{X}$. We can write this binarization β as a sequence of reductions, where each reduction is characterized by a pair of position sets (X_1, X_2) which are merged into $X_1 \cup X_2$, in such a way that both each of the initial sets and the result of the merge have fan-out at most 2.

We will show that, under these conditions, for every pair of adjacent position sets D_1 and D_2, there exists a binarization that starts with the reduction merging D_1 with D_2.

Without loss of generality, we assume that $f(D_1) \leq f(D_2)$ (if this inequality does not hold we can always swap the names of the two position sets, since the merging operation is commutative), and we define a function $h_{D_1 \rightarrow D_2} : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$ as follows:

- $h_{D_1 \rightarrow D_2}(X) = X$; if $D_1 \nsubseteq X \wedge D_2 \nsubseteq X$.
- $h_{D_1 \rightarrow D_2}(X) = X$; if $D_1 \subseteq X \wedge D_2 \subseteq X$.
- $h_{D_1 \rightarrow D_2}(X) = X \cup D_1$; if $D_1 \nsubseteq X \wedge D_2 \subseteq X$.
- $h_{D_1 \rightarrow D_2}(X) = X \setminus D_1$; if $D_1 \subseteq X \wedge D_2 \nsubseteq X$.

With this, we construct a binarization β' from β as follows:

- The first reduction in β' merges the pair of position sets (D_1, D_2),
- We consider the reductions in β in order, and for each reduction o merging (X_1, X_2), if $X_1 \neq D_1$ and $X_2 \neq D_1$, we append a reduction o' merging $(h_{D_1 \rightarrow D_2}(X_1), h_{D_1 \rightarrow D_2}(X_2))$ to β'.

We will now prove that, if β is a 2-feasible binarization, then β' is also a 2-feasible binarization. To prove this, it suffices to show the following:[2]

(i) Every position set merged by a reduction in β' is either one of the original sets in $\mathcal{X}$, or the result of a previous merge in β'.

(ii) Every reduction in β' merges a pair of position sets (X_1, X_2) which are 2-combinable.

To prove (i) we note that by construction of β', if an operand of a merging operation in β' is not one of the original position sets in $\mathcal{X}$, then it must be an $h_{D_1 \rightarrow D_2}(X)$ for some X that appears as an operand of a merging operation in β. Since the binarization β is itself valid, this X must be either one of the position sets in $\mathcal{X}$, or the result of a previous merge in the binarization β. So we divide the proof into two cases:

- If $X \in \mathcal{X}$: First of all, we note that X cannot be D_1, since the merging operations of β that have D_1 as an operand do not produce

[2] It is also necessary to show that no position set is merged in two different reductions, but this easily follows from the fact that $h_{D_1 \rightarrow D_2}(X) = h_{D_1 \rightarrow D_2}(Y)$ if and only if $X \cup D_1 = Y \cup D_1$. Thus, two reductions in β can only produce conflicting reductions in β' if they merge two position sets differing only by D_1, but in this case, one of the reductions must merge D_1 so it does not produce any reduction in β'.

a corresponding operation in β'. If X equals D_2, then $h_{D_1 \to D_2}(X)$ is $D_1 \cup D_2$, which is the result of the first merging operation in β'. Finally, if X is one of the position sets in $\mathcal{X}$, and not D_1 or D_2, then $h_{D_1 \to D_2}(X) = X$, so our operand is also one of the position sets in $\mathcal{X}$.

- If X is the result of a previous merging operation o in binarization β: Then, $h_{D_1 \to D_2}(X)$ is the result of a previous merging operation o' in binarization β', which is obtained by applying the function $h_{D_1 \to D_2}$ to the operands and result of o. [3]

To prove (ii), we show that, under the assumptions of the theorem, the function $h_{D_1 \to D_2}$ preserves 2-combinability. Since two position sets of fan-out ≤ 2 are 2-combinable if and only if they are disjoint and the fan-out of their union is at most 2, it suffices to show that, for every X, X_1, X_2 unions of one or more sets of $\mathcal{X}$, having fan-out ≤ 2, such that $X_1 \neq D_1$, $X_2 \neq D_1$ and $X \neq D_1$;

(a) The function $h_{D_1 \to D_2}$ preserves disjointness, that is, if X_1 and X_2 are disjoint, then $h_{D_1 \to D_2}(X_1)$ and $h_{D_1 \to D_2}(X_2)$ are disjoint.

(b) The function $h_{D_1 \to D_2}$ is distributive with respect to the union of position sets, that is, $h_{D_1 \to D_2}(X_1 \cup X_2) = h_{D_1 \to D_2}(X_1) \cup h_{D_1 \to D_2}(X_2)$.

(c) The function $h_{D_1 \to D_2}$ preserves the property of having fan-out ≤ 2, that is, if X has fan-out ≤ 2, then $h_{D_1 \to D_2}(X)$ has fan-out ≤ 2.

If X_1 and X_2 do not contain D_1 or D_2, or if one of the two unions X_1 or X_2 contains $D_1 \cup D_2$, properties (a) and (b) are trivial, since the function $h_{D_1 \to D_2}$ behaves as the identity function in these cases.

It remains to show that (a) and (b) are true in the following cases:

- X_1 contains D_1 but not D_2, and X_2 does not contain D_1 or D_2:

  In this case, if X_1 and X_2 are disjoint, we can write $X_1 = Y_1 \cup D_1$, such that Y_1, X_2, D_1 are pairwise disjoint. By definition, we have that $h_{D_1 \to D_2}(X_1) = Y_1$, and $h_{D_1 \to D_2}(X_2) = X_2$, which are disjoint, so (a) holds.

  Property (b) also holds because, with these expressions for X_1 and X_2, we can calculate $h_{D_1 \to D_2}(X_1 \cup X_2) = Y_1 \cup X_2 = h_{D_1 \to D_2}(X_1) \cup h_{D_1 \to D_2}(X_2)$.

- X_1 contains D_2 but not D_1, X_2 does not contain D_1 or D_2:

  In this case, if X_1 and X_2 are disjoint, we can write $X_1 = Y_1 \cup D_2$, such that Y_1, X_2, D_1, D_2 are pairwise disjoint. By definition, $h_{D_1 \to D_2}(X_1) = Y_1 \cup D_2 \cup D_1$, and $h_{D_1 \to D_2}(X_2) = X_2$, which are disjoint, so (a) holds.

  Property (b) also holds, since we can check that $h_{D_1 \to D_2}(X_1 \cup X_2) = Y_1 \cup X_2 \cup D_2 \cup D_1 = h_{D_1 \to D_2}(X_1) \cup h_{D_1 \to D_2}(X_2)$.

- X_1 contains D_1 but not D_2, X_2 contains D_2 but not D_1:

  In this case, if X_1 and X_2 are disjoint, we can write $X_1 = Y_1 \cup D_1$ and $X_2 = Y_2 \cup D_2$, such that Y_1, Y_2, D_1, D_2 are pairwise disjoint. By definition, we know that $h_{D_1 \to D_2}(X_1) = Y_1$, and $h_{D_1 \to D_2}(X_2) = Y_2 \cup D_1 \cup D_2$, which are disjoint, so (a) holds.

  Finally, property (b) also holds in this case, since $h_{D_1 \to D_2}(X_1 \cup X_2) = Y_1 \cup X_2 \cup D_2 \cup D_1 = h_{D_1 \to D_2}(X_1) \cup h_{D_1 \to D_2}(X_2)$.

This concludes the proof of (a) and (b).

To prove (c), we consider a position set X, union of one or more sets of $\mathcal{X}$, with fan-out ≤ 2 and such that $X \neq D_1$. First of all, we observe that if X does not contain D_1 or D_2, or if it contains $D_1 \cup D_2$, (c) is trivial, because the function $h_{D_1 \to D_2}$ behaves as the identity function in this case. So it remains to prove (c) in the cases where X contains D_1 but not D_2, and where X contains D_2 but not D_1. In any of these two cases, if we call $E(Y)$ the endpoint set associated with an arbitrary position set Y, we can make the following observations:

1. Since X has fan-out ≤ 2, $E(X)$ contains at most 4 endpoints.

2. Since D_1 has fan-out $f(D_1)$, $E(D_1)$ contains at most $2f(D_1)$ endpoints.

[3] Except if one of the operands of the operation o was D_1. But in this case, if we call the other operand Z, then we have that $X = D_1 \cup Z$. If Z contains D_2, then $X = D_1 \cup Z = h_{D_1 \to D_2}(X) = h_{D_1 \to D_2}(Z)$, so we apply this same reasoning with $h_{D_1 \to D_2}(Z)$ where we cannot fall into this case, since there can be only one merge operation in β that uses D_1 as an operand. If Z does not contain D_2, then we have that $h_{D_1 \to D_2}(X) = X \setminus D_1 = Z = h_{D_1 \to D_2}(Z)$, so we can do the same.

3. Since D_2 has fan-out $f(D_2)$, $E(D_2)$ contains at most $2f(D_2)$ endpoints.

4. Since D_1 and D_2 are adjacent, we know that $E(D_1) \cap E(D_2)$ contains at *least* $min(f(D_1), f(D_2)) = f(D_1)$ endpoints.

5. Therefore, $E(D_1) \setminus (E(D_1) \cap E(D_2))$ can contain at most $2f(D_1) - f(D_1) = f(D_1)$ endpoints.

6. On the other hand, since X contains only one of D_1 and D_2, we know that the endpoints where D_1 is adjacent to D_2 must also be endpoints of X, so that $E(D_1) \cap E(D_2) \subseteq E(X)$. Therefore, $E(X) \setminus (E(D_1) \cap E(D_2))$ can contain at most $4 - f(D_1)$ endpoints.

Now, in the case where X contains D_1 but not D_2, we know that $h_{D_1 \to D_2}(X) = X \setminus D_1$. We calculate a bound for the fan-out of $X \setminus D_1$ as follows: we observe that all the endpoints in $E(X \setminus D_1)$ must be either endpoints of X or endpoints of D_1, since $E(X) = (E(X \setminus D_1) \cup E(D_1)) \setminus (E(X \setminus D_1) \cap E(D_1))$, so every position that is in $E(X \setminus D_1)$ but not in $E(D_1)$ must be in $E(X)$. But we also observe that $E(X \setminus D_1)$ cannot contain any of the endpoints where D_1 is adjacent to D_2 (i.e., the members of $E(D_1) \cap E(D_2)$), since $X \setminus D_1$ does not contain D_1 or D_2. Thus, we can say that any endpoint of $X \setminus D_1$ is either a member of $E(D_1) \setminus (E(D_1) \cap E(D_2))$, or a member of $E(X) \setminus (E(D_1) \cap E(D_2))$.

Thus, the number of endpoints in $E(X \setminus D_1)$ cannot exceed the sum of the number of endpoints in these two sets, which, according to the reasonings above, is at most $4 - f(D_1) + f(D_1) = 4$. Since $E(X \setminus D_1)$ cannot contain more than 4 endpoints, we conclude that the fan-out of $X \setminus D_1$ is at most 2, so the function $h_{D_1 \to D_2}$ preserves the property of position sets having fan-out ≤ 2 in this case.

In the other case, where X contains D_2 but not D_1, we follow a similar reasoning: in this case, $h_{D_1 \to D_2}(X) = X \cup D_1$. To bound the fan-out of $X \cup D_1$, we observe that all the endpoints in $E(X \cup D_1)$ must be either in $E(X)$ or in $E(D_1)$, since $E(X \cup D_1) = (E(X) \cup E(D_1)) \setminus (E(X) \cap E(D_1))$. But we also know that $E(X \cup D_1)$ cannot contain any of the endpoints where D_1 is adjacent to D_2 (i.e., the members of $E(D_1) \cap E(D_2)$), since $X \cup D_1$ contains both D_1 and D_2. Thus, we can say that any endpoint of $X \cup D_1$ is either a member of $E(D_1) \setminus (E(D_1) \cap E(D_2))$, or a member of $E(X) \setminus (E(D_1) \cap E(D_2))$. Reasoning as in the previous case, we conclude that the fan-out of $X \cup D_1$ is at most 2, so the function $h_{D_1 \to D_2}$ also preserves the property of position sets having fan-out ≤ 2 in this case.

This concludes the proof of Theorem 1.

```
1: Function BINARIZATION(p)
2: A ← ∅; {working agenda}
3: R ← ⟨⟩; {empty list of reductions}
4: for all i from 1 to r(p) do
5:     A ← A ∪ {X_{A_i}};
6: while |A| > 2 and A contains two adjacent
   position sets do
7:     choose X_1, X_2 ∈ A such that X_1 ↔ X_2;
8:     X ← X_1 ∪ X_2;
9:     A ← (A \ {X_1, X_2}) ∪ {X};
10:    append (X_1, X_2) to R;
11: if |A| = 2 then
12:    return R;
13: else
14:    return fail;
```

Figure 1: Binarization algorithm for a production $p : A \to g(A_1, \ldots, A_{r(p)})$. Result is either a list of reductions or failure.

4 Binarization algorithm

Let $p : A \to g(A_1, \ldots, A_{r(p)})$ be a production with $r(p) > 2$ from some LCFRS with fan-out not greater than 2. Recall from Subsection 3.1 that each occurrence of nonterminal A_i in the right-hand side of p is represented as a position set X_{A_i}. The specification of an algorithm for finding a 2-feasible binarization of p is reported in Figure 1.

The algorithm uses an agenda $\mathcal{A}$ as a working set, where all position sets that still need to be processed are stored. $\mathcal{A}$ is initialized with the position sets X_{A_i}, $1 \leq i \leq r(p)$. At each step in the algorithm, the size of $\mathcal{A}$ represents the maximum rank among all productions that can be obtained from the reductions that have been chosen so far in the binarization process. The algorithm also uses a list $\mathcal{R}$, initialized as the empty list, where all reductions that are attempted in the binarization process are appended.

At each iteration, the algorithm performs a reduction by arbitrarily choosing a pair of adjacent endpoint sets from the agenda and by merging them. As already discussed in Subsection 3.1, this

corresponds to some specific transformation of the input production p that preserves its generative capacity and that decreases its rank by one unit.

We stop the iterations of the algorithm when we reach a state in which there are no more than two position sets in the agenda. This means that the binarization process has come to an end with the reduction of p to a set of productions equivalent to p and with rank and fan-out at most 2. This set of productions can be easily constructed from the output list $\mathcal{R}$. We also stop the iterations in case no adjacent pair of position sets can be found in the agenda. If the agenda has more than two position sets, this means that no binarization has been found and the algorithm returns a failure.

4.1 Correctness

To prove the correctness of the algorithm in Figure 1, we need to show that it produces a 2-feasible binarization of the given production p whenever such a binarization exists. This is established by the following theorem:

Theorem 2 *Let $\mathcal{X}$ be a 2-feasible collection of position sets, such that the union of all sets in $\mathcal{X}$ is a position set with fan-out ≤ 2. The procedure:*

while ($\mathcal{X}$ contains any pair of adjacent sets X_1, X_2) reduce $\mathcal{X}$ by merging X_1 with X_2;

always finds a 2-feasible binarization of $\mathcal{X}$.

In order to prove this, the loop invariant is that $\mathcal{X}$ is a 2-feasible set, and that the union of all position sets in $\mathcal{X}$ has fan-out ≤ 2: reductions can never change the union of all sets in $\mathcal{X}$, and Theorem 1 guarantees us that every change to the state of $\mathcal{X}$ maintains 2-feasibility. We also know that the algorithm eventually finishes, because every iteration reduces the amount of position sets in $\mathcal{X}$ by 1; and the looping condition will not hold when the number of sets gets to be 1.

So it only remains to prove that the loop is only exited if $\mathcal{X}$ contains at most two position sets. If we show this, we know that the sequence of reductions produced by this procedure is a 2-feasible binarization. Since the loop is exited when $\mathcal{X}$ is 2-feasible but it contains no pair of adjacent position sets, it suffices to show the following:

Proposition 1 *Let $\mathcal{X}$ be a 2-feasible collection of position sets, such that the union of all the sets in $\mathcal{X}$ is a position set with fan-out ≤ 2. If $\mathcal{X}$ has more than two elements, then it contains at least a pair of adjacent position sets.* □

Let $\mathcal{X}$ be a 2-feasible collection of more than two position sets. Since $\mathcal{X}$ is 2-feasible, we know that there must be a 2-feasible binarization of $\mathcal{X}$. Suppose that β is such a binarization, and let D_1 and D_2 be the two position sets that are merged in the first reduction of β. Since β is 2-feasible, D_1 and D_2 must be 2-combinable.

If D_1 and D_2 are adjacent, our proposition is true. If they are not adjacent, then, in order to be 2-combinable, the fan-out of both position sets must be 1: if any of them had fan-out 2, their union would need to have fan-out > 2 for D_1 and D_2 not to be adjacent, and thus they would not be 2-combinable. Since D_1 and D_2 have fan-out 1 and are not adjacent, their sets of endpoints are of the form $\{b_1, b_2\}$ and $\{c_1, c_2\}$, and they are disjoint.

If we call $E_{\mathcal{X}}$ the set of endpoints corresponding to the union of all the position sets in $\mathcal{X}$ and $E_{D_1 D_2} = \{b_1, b_2, c_1, c_2\}$, we can show that at least one of the endpoints in $E_{D_1 D_2}$ does not appear in $E_{\mathcal{X}}$, since we know that $E_{\mathcal{X}}$ can have at most 4 elements (as the union has fan-out ≤ 2) and that it cannot equal $E_{D_1 D_2}$ because this would mean that $\mathcal{X} = \{D_1, D_2\}$, and by hypothesis $\mathcal{X}$ has more than two position sets. If we call this endpoint x, this means that there must be a position set D_3 in $\mathcal{X}$, different from D_1 and D_2, that has x as one of its endpoints. Since D_1 and D_2 have fan-out 1, this implies that D_3 must be adjacent either to D_1 or to D_2, so we conclude the proof.

4.2 Implementation and complexity

We now turn to the computational analysis of the algorithm in Figure 1. We define the **length** of an LCFRS production p, written $|p|$, as the sum of the length of all strings α_j in $\vec{\alpha}$ in the definition of the linear, non-erasing function associated with p. Since we are dealing with LCFRS of fan-out at most two, we easily derive that $|p| = \mathcal{O}(r(p))$.

In the implementation of the algorithm it is convenient to represent each position set by means of the corresponding endpoint set. Since at any time in the computation we are only processing position sets with fan-out not greater than two, each endpoint set will contain at most four integers.

The for-loop at lines 4 and 5 in the algorithm can be easily implemented through a left-to-right scan of the characteristic string $\sigma_N(p)$, detecting the endpoint sets associated with each position set X_{A_i}. This can be done in constant time for each

X_{A_i}, and thus in linear time in $|p|$.

At each iteration of the while-loop at lines 6 to 10 we have that $\mathcal{A}$ is reduced in size by one unit. This means that the number of iterations is bounded by $r(p)$. We will show below that each iteration of this loop can be executed in constant time. We can therefore conclude that our binarization algorithm runs in optimal time $\mathcal{O}(|p|)$.

In order to run in constant time each single iteration of the while-loop at lines 6 to 10, we need to perform some additional bookkeeping. We use two arrays V_e and V_a, whose elements are indexed by the endpoints associated with characteristic string $\sigma_N(p)$, that is, integers $i \in [0, |\sigma_N(p)|]$. For each endpoint i, $V_e[i]$ stores all the endpoint sets that share endpoint i. Since each endpoint can be shared by at most two endpoint sets, such a data structure has size $\mathcal{O}(|p|)$. If there exists some position set X in $\mathcal{A}$ with leftmost endpoint i, then $V_a[i]$ stores all the position sets (represented as endpoint sets) that are adjacent to X. Since each position set can be adjacent to at most four other position sets, such a data structure has size $\mathcal{O}(|p|)$. Finally, we assume we can go back and forth between position sets in the agenda and their leftmost endpoints.

We maintain arrays V_e and V_a through the following simple procedures.

- Whenever a new position set X is added to $\mathcal{A}$, for each endpoint i of X we add X to $V_e[i]$. We also check whether any position set in $V_e[i]$ other than X is adjacent to X, and add these position sets to $V_a[i_l]$, where i_l is the leftmost end point of X.
- Whenever some position set X is removed from $\mathcal{A}$, for each endpoint i of X we remove X from $V_e[i]$. We also remove all of the position sets in $V_a[i_l]$, where i_l is the leftmost end point of X.

It is easy to see that, for any position set X which is added/removed from $\mathcal{A}$, each of the above procedures can be executed in constant time.

We maintain a set $\mathcal{I}$ of integer numbers $i \in [0, |\sigma_N(p)|]$ such that $i \in \mathcal{I}$ if and only if $V_a[i]$ is not empty. Then at each iteration of the while-loop at lines 6 to 10 we pick up some index in $\mathcal{I}$ and retrieve at $V_a[i]$ some pair X, X' such that $X \leftrightarrow X'$. Since X, X' are represented by means of endpoint sets, we can compute the endpoint set of $X \cup X'$ in constant time. Removal of X, X' and addition of $X \cup X'$ in our data structures V_e and V_a is then performed in constant time, as described above. This proves our claim that each single iteration of the while loop can be executed in constant time.

5 Discussion

We have presented an algorithm for the binarization of a LCFRS with fan-out 2 that does not increase the fan-out, and have discussed how this can be applied to improve parsing efficiency in several practical applications. In the algorithm of Figure 1, we can modify line 14 to return $\mathcal{R}$ even in case of failure. If we do this, when a binarization with fan-out ≤ 2 does not exist the algorithm will still provide us with a list of reductions that can be converted into a set of productions equivalent to p with fan-out at most 2 and rank bounded by some r_b, with $2 < r_b \leq r(p)$. In case $r_b < r(p)$, we are not guaranteed to have achieved an optimal reduction in the rank, but we can still obtain an asymptotic improvement in parsing time if we use the new productions obtained in the transformation.

Our algorithm has optimal time complexity, since it works in linear time with respect to the input production length. It still needs to be investigated whether the proposed technique, based on determinization of the choice of the reduction, can also be used for finding binarizations for LCFRS with fan-out larger than two, again without increasing the fan-out. However, it seems unlikely that this can still be done in linear time, since the problem of binarization for LCFRS in general, i.e., without any bound on the fan-out, might not be solvable in polynomial time. This is still an open problem; see (Gómez-Rodríguez et al., 2009) for discussion.

Acknowledgments

The first author has been supported by Ministerio de Educación y Ciencia and FEDER (HUM2007-66607-C04) and Xunta de Galicia (PGIDIT-07SIN005206PR, INCITE08E1R104022ES, INCITE08ENA305025ES, INCITE08PXIB-302179PR and Rede Galega de Procesamento da Linguaxe e Recuperación de Información). The second author has been partially supported by MIUR under project PRIN No. 2007TJNZRE_002.

References

Pierre Boullier. 2004. Range concatenation grammars. In H. Bunt, J. Carroll, and G. Satta, editors, *New Developments in Parsing Technology*, volume 23 of *Text, Speech and Language Technology*, pages 269–289. Kluwer Academic Publishers.

Håkan Burden and Peter Ljunglöf. 2005. Parsing linear context-free rewriting systems. In *IWPT05, 9th International Workshop on Parsing Technologies*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd ACL*, pages 263–270.

Carlos Gómez-Rodríguez, Marco Kuhlmann, Giorgio Satta, and David Weir. 2009. Optimal reduction of rule length in linear context-free rewriting systems. In *Proc. of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies Conference (NAACL'09:HLT)*, Boulder, Colorado. To appear.

Aravind K. Joshi and Leon S. Levy. 1977. Constraints on local descriptions: Local transformations. *SIAM J. Comput.*, 6(2):272–284.

Aravind K. Joshi, K. Vijay-Shanker, and David Weir. 1991. The convergence of mildly context-sensitive grammatical formalisms. In P. Sells, S. Shieber, and T. Wasow, editors, *Foundational Issues in Natural Language Processing*. MIT Press, Cambridge MA.

Marco Kuhlmann and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proc. of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, pages 478–486, Athens, Greece.

I. Dan Melamed. 2003. Multitext grammars and synchronous parsers. In *Proceedings of HLT-NAACL 2003*.

Rebecca Nesson and Stuart M. Shieber. 2006. Simpler TAG semantics through synchronization. In *Proceedings of the 11th Conference on Formal Grammar*, Malaga, Spain, 29–30 July.

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

Giorgio Satta. 1998. Trading independent for synchronized parallelism in finite copying parallel rewriting systems. *Journal of Computer and System Sciences*, 56(1):27–45.

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

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Meeting of the Association for Computational Linguistics (ACL'87)*.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting synchronous grammar rules from word-level alignments in linear time. In *22nd International Conference on Computational Linguistics (Coling)*, pages 1081–1088, Manchester, England, UK.