Optimal rank reduction for Linear Context-Free Rewriting Systems with Fan-Out Two

Benot Sagot
INRIA & Université Paris 7
Le Chesnay, France
benoit.sagot@inria.fr

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

Abstract

Linear Context-Free Rewriting Systems (LCFRSs) are a grammar formalism capable of modeling discontinuous phrases. Many parsing applications use LCFRSs where the *fan-out* (a measure of the discontinuity of phrases) does not exceed 2. We present an efficient algorithm for optimal reduction of the length of production right-hand side in LCFRSs with fan-out at most 2. This results in asymptotical running time improvement for known parsing algorithms for this class.

1 Introduction

Linear Context-Free Rewriting Systems (LCFRSs) have been introduced by Vijay-Shanker *et al.* (1987) for modeling the syntax of natural language. The formalism extends the generative capacity of context-free grammars, still remaining far below the class of context-sensitive grammars. An important feature of LCFRSs is their ability to generate discontinuous phrases. This has been recently exploited for modeling phrase structure treebanks with discontinuous constituents (Maier and Søgaard, 2008), as well as non-projective dependency treebanks (Kuhlmann and Satta, 2009).

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) can also be viewed as a special kind of LCFRS with $f = 2$, since each auxiliary tree generates two strings, and with r given by the maximum number of adjunction and substitution sites in an elementary tree. Beyond tree adjoining languages, LCFRSs with $f = 2$ can also generate languages in which pair of strings derived from different nonterminals appear in so-called crossing configurations. It has recently been observed that, in this way, LCFRSs with $f = 2$ can model the vast majority of data in discontinuous phrase structure treebanks and non-projective dependency treebanks (Maier and Lichte, 2009; Kuhlmann and Satta, 2009).

Under a theoretical perspective, the parsing problem for LCFRSs with $f = 2$ is NP-complete (Satta, 1992), and in known parsing algorithms the running time is exponentially affected by the rank r of the grammar. Nonetheless, in natural language parsing applications, it is possible to achieve efficient, polynomial parsing if we succeed in reducing the rank r (number of nonterminals in the right-hand side) of individual LCFRSs' productions (Kuhlmann and Satta, 2009). This process is called production **factorization**. Production factorization is very similar to the reduction of a context-free grammar production into Chomsky normal form. However, in the LCFRS case some productions might not be reducible to $r = 2$, and the process stops at some larger value for r, which in the worst case might as well be the rank of the source production (Rambow and Satta, 1999).

Motivated by parsing efficiency, the factorization problem for LCFRSs with $f = 2$ has attracted the attention of many researchers in recent years. Most of the literature has been focusing on binarization algorithms, which attempt to find a reduction to $r = 2$ and return a failure if this is not possible. Gómez-Rodríguez *et al.* (2009) report a general binarization algorithm for LCFRS which, in the case of $f = 2$, works in time $\mathcal{O}(|p|^7)$, where $|p|$ is the size of the input production. A more efficient binarization algorithm for the case $f = 2$ is presented in (Gómez-Rodríguez and Satta, 2009), working in time $\mathcal{O}(|p|)$.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 525–533,
Uppsala, Sweden, 11-16 July 2010.

In this paper we are interested in general factorization algorithms, i.e., algorithms that find factorizations with the smallest possible rank (not necessarily $r = 2$). We present a novel technique that solves the general factorization problem in time $\mathcal{O}(|p|^2)$ for LCFRSs with $f = 2$.

Strong generative equivalence results between LCFRS and other finite copying parallel rewriting systems have been discussed in (Weir, 1992) and in (Rambow and Satta, 1999). Through these equivalence results, we can transfer the factorization techniques presented in this article to other finite copying parallel rewriting systems.

2 LCFRSs

In this section we introduce the basic notation for LCFRS and the notion of production factorization.

2.1 Definitions

Let Σ_T be a finite alphabet of terminal symbols. As usual, Σ_T^* denotes the set of all finite strings over Σ_T, including the empty string ε. For integer $k \geq 1$, $(\Sigma_T^*)^k$ denotes the set of all tuples $(w_1, \ldots, w_k)$ of strings $w_i \in \Sigma_T^*$. In what follows we are interested in functions mapping several tuples of strings in Σ_T^* into tuples of strings in Σ_T^*. Let r and f be two integers, $r \geq 0$ and $f \geq 1$. We say that a function g has **rank** r if there exist integers $f_i \geq 1$, $1 \leq i \leq r$, such that g is defined on $(\Sigma_T^*)^{f_1} \times (\Sigma_T^*)^{f_2} \times \cdots \times (\Sigma_T^*)^{f_r}$. We also say that g has **fan-out** f if the range of g is a subset of $(\Sigma_T^*)^f$. Let y_h, x_{ij}, $1 \leq h \leq f$, $1 \leq i \leq r$ and $1 \leq j \leq f_i$, be string-valued variables. A function g as above is said to be **linear regular** if it is defined by an equation of the form

$$g(\langle x_{11}, \ldots, x_{1f_1} \rangle, \ldots, \langle x_{r1}, \ldots, x_{rf_r} \rangle) = \\ = \langle y_1, \ldots, y_f \rangle, \quad (1)$$

where $\langle y_1, \ldots, y_f \rangle$ represents some grouping into f sequences of all and only the variables appearing in the left-hand side of (1) (without repetitions) along with some additional terminal symbols (with possible repetitions).

For a mathematical definition of LCFRS we refer the reader to (Weir, 1992, p. 137). Informally, in a LCFRS every nonterminal symbol A is associated with an integer $\varphi(A) \geq 1$, called its fan-out, and it generates tuples in $(\Sigma_T^*)^{\varphi(A)}$. Productions in a LCFRS have the form

$$p: \quad A \rightarrow g(B_1, B_2, \ldots, B_{\rho(p)}),$$

where $\rho(p) \geq 0$, A and B_i, $1 \leq i \leq \rho(p)$, are nonterminal symbols, and g is a linear regular function having rank $\rho(p)$ and fan-out $\varphi(A)$, defined on $(\Sigma_T^*)^{\varphi(B_1)} \times \cdots \times (\Sigma_T^*)^{\varphi(B_{\rho(p)})}$ and taking values in $(\Sigma_T^*)^{\varphi(A)}$. The basic idea underlying the rewriting relation associated with LCFRS is that production p applies to any sequence of string tuples generated by the B_i's, and provides a new string tuple in $(\Sigma_T^*)^{\varphi(A)}$ obtained through function g. We say that $\varphi(p) = \varphi(A)$ is the **fan-out** of p, and $\rho(p)$ is the **rank** of p.

Example 1 Let L be the language $L = \{a^n b^n a^m b^m a^n b^n a^m b^m \mid n, m \geq 1\}$. A LCFRS generating L is defined by means of the nonterminals S, $\varphi(S) = 1$, and A, $\varphi(A) = 2$, and the productions in figure 1. Observe that nonterminal A generates all tuples of the form $\langle a^n b^n, a^n b^n \rangle$. □

Recognition and parsing for a given LCFRS can be carried out in polynomial time on the length of the input string. This is usually done by exploiting standard dynamic programming techniques; see for instance (Seki et al., 1991).[1] However, the polynomial degree in the running time is a monotonically strictly increasing function that depends on both the rank and the fan-out of the productions in the grammar. To optimize running time, one can then recast the source grammar in such a way that the value of the above function is kept to a minimum. One way to achieve this is by factorizing the productions of a LCFRS, as we now explain.

2.2 Factorization

Consider a LCFRS production of the form $p: \quad A \rightarrow g(B_1, B_2, \ldots, B_{\rho(p)})$, where g is specified as in (1). Let also $\mathcal{C}$ be a subset of $\{B_1, B_2, \ldots, B_{\rho(p)}\}$ such that $|\mathcal{C}| \neq 0$ and $|\mathcal{C}| \neq \rho(p)$. We let $\Sigma_{\mathcal{C}}$ be the alphabet of all variables x_{ij} defined as in (1), for all values of i and j such that $B_i \in \mathcal{C}$ and $1 \leq j \leq f_i$. For each i with $1 \leq i \leq f$, we rewrite each string y_i in (1) in a form $y_i = y'_{i0} z_{i1} y'_{i1} \cdots y'_{id_{i-1}} z_{id_i} y'_{id_i}$, with $d_i \geq 0$, such that the following conditions are all met:

- each z_{ij}, $1 \leq j \leq d_i$, is a string with one or more occurrences of variables, all in $\Sigma_{\mathcal{C}}$;
- each y'_{ij}, $1 \leq j \leq d_i - 1$, is a non-empty string with no occurrences of symbols in $\Sigma_{\mathcal{C}}$;
- y'_{0j} and y'_{0d_i} are (possibly empty) strings with no occurrences of symbols in $\Sigma_{\mathcal{C}}$.

[1] In (Seki et al., 1991) a syntactic variant of LCFRS is used, called multiple context-free grammars.

$$
\begin{array}{ll}
S \rightarrow g_S(A, A), & g_S(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle) = \langle x_{11}x_{21}x_{12}x_{22}\rangle; \\
A \rightarrow g_A(A), & g_A(\langle x_{11}, x_{12}\rangle) = \langle ax_{11}b, ax_{12}b\rangle; \\
A \rightarrow g'_A(), & g'_A() = \langle ab, ab\rangle.
\end{array}
$$

Figure 1: A LCFRS for language $L = \{a^n b^n a^m b^m a^n b^n a^m b^m \mid n, m \geq 1\}$.

Let $c = |\mathcal{C}|$ and $\overline{c} = \rho(p) - |\mathcal{C}|$. Assume that $\mathcal{C} = \{B_{h_1}, \ldots, B_{h_c}\}$, and $\{B_1, \ldots, B_{\rho(p)}\} - \mathcal{C} = \{B_{h'_1}, \ldots, B_{h'_{\overline{c}}}\}$. We introduce a fresh nonterminal C with $\varphi(C) = \sum_{i=1}^{f} d_i$ and replace production p in our grammar by means of the two new productions $p_1 : C \rightarrow g_1(B_{h_1}, \ldots, B_{h_c})$ and $p_2 : A \rightarrow g_2(C, B_{h'_1}, \ldots, B_{h'_{\overline{c}}})$. Functions g_1 and g_2 are defined as:

$$
\begin{array}{l}
g_1(\langle x_{h_1 1}, \ldots, x_{h_1 f_{h_1}}\rangle, \ldots, \langle x_{h_c 1}, \ldots, x_{h_c f_{h_c}}\rangle) \\
\quad = \langle z_{11}, \cdots, z_{1d_1}, z_{21}, \cdots, z_{fd_f}\rangle; \\
g_2(\langle x_{h'_1 1}, \ldots, x_{h'_1 f_{h'_1}}\rangle, \ldots, \langle x_{h'_{\overline{c}} 1}, \ldots, x_{h'_{\overline{c}} f_{h'_{\overline{c}}}}\rangle) \\
\quad = \langle y'_{10}, \ldots, y'_{1d_1}, y'_{20}, \ldots, y'_{fd_f}\rangle.
\end{array}
$$

Note that productions p_1 and p_2 have rank strictly smaller than the source production p. Furthermore, if it is possible to choose set $\mathcal{C}$ in such a way that $\sum_{i=0}^{f} d_i \leq f$, then the fan-out of p_1 and p_2 will be no greater than the fan-out of p.

We can iterate the procedure above as many times as possible, under the condition that the fan-out of the productions does not increase.

Example 2 Let us consider the following production with rank 4:

$$
\begin{array}{l}
A \rightarrow g_S(B, C, D, E), \\
\quad g_A(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle, \langle x_{31}, x_{32}\rangle, \langle x_{41}, x_{42}\rangle) \\
\quad\quad = \langle x_{11}x_{21}x_{31}x_{41}x_{12}x_{42}, x_{22}x_{32}\rangle.
\end{array}
$$

Applyng the above procedure twice, we obtain a factorization consisting of three productions with rank 2 (variables have been renamed to reflect our conventions):

$$
\begin{array}{l}
A \rightarrow g_A(A_1, A_2), \\
\quad g_A(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle) \\
\quad\quad = \langle x_{11}x_{21}x_{12}, x_{22}\rangle; \\
A_1 \rightarrow g_{A_1}(B, E), \\
\quad g_{A_1}(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle) = \langle x_{11}, x_{21}x_{12}x_{22}\rangle; \\
A_2 \rightarrow g_{A_2}(C, D), \\
\quad g_{A_2}(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle) = \langle x_{11}x_{21}, x_{12}x_{22}\rangle.
\end{array}
$$

□

The factorization procedure above should be applied to all productions of a LCFRS with rank larger than two. This might result in an asymptotic improvement of the running time of existing dynamic programming algorithms for parsing based on LCFRS.

The factorization technique we have discussed can also be viewed as a generalization of well-known techniques for casting context-free grammars into binary forms. These are forms where no more than two nonterminal symbols are found in the right-hand side of productions of the grammar; see for instance (Harrison, 1978). One important difference is that, while production factorization into binary form is always possible in the context-free case, for LCFRS there are worst case grammars in which rank reduction is not possible at all, as shown in (Rambow and Satta, 1999).

3 A graph-based representation for LCFRS productions

Rather than factorizing LCFRS productions directly, in this article we work with a more abstract representation of productions based on graphs. From now on we focus on LCFRS whose nonterminals and productions all have fan-out smaller than or equal to 2. Consider then a production $p : A \rightarrow g(B_1, B_2, \ldots, B_{\rho(p)})$, with $\varphi(A), \varphi(B_i) \leq 2$, $1 \leq i \leq \rho(p)$, and with g defined as

$$
\begin{array}{l}
g(\langle x_{11}, \ldots, x_{1\varphi(B_1)}\rangle, \ldots \\
\quad \ldots, \langle x_{\rho(p)1}, \ldots, x_{\rho(p)\varphi(B_{\rho(p)})}\rangle) \\
\quad = \langle y_1, \ldots, y_{\varphi(A)}\rangle.
\end{array}
$$

In what follows, if $\varphi(A) = 1$ then $\langle y_1, \ldots, y_{\varphi(A)}\rangle$ should be read as $\langle y_1\rangle$ and $y_1 \cdots y_{\varphi(A)}$ should be read as y_1. The same convention applies to all other nonterminals and tuples.

We now introduce a special kind of undirected graph that is associated with a linear order defined over the set of its vertices. The **p-graph** associated with production p is a triple $(V_p, E_p, \prec_p)$ such that

- $V_p = \{x_{ij} \mid 1 \leq i \leq \rho(p),\ \varphi(B_i) = 2,\ 1 \leq j \leq \varphi(B_i)\}$ is a set of vertices;[2]

[2]Here we are overloading symbols x_{ij}. It will always be clear from the context whether x_{ij} is a string-valued variable or a vertex in a p-graph.

- $E_p = \{(x_{i1}, x_{i2}) \mid x_{i1}, x_{i2} \in V_p\}$ is a set of undirected edges;
- for $x, x' \in V_p$, $x \prec_p x'$ if $x \neq x'$ and the (unique) occurrence of x in $y_1 \cdots y_{\varphi(A)}$ precedes the (unique) occurrence of x'.

Note that in the above definition we are ignoring all string-valued variables x_{ij} associated with nonterminals B_i with $\varphi(B_i) = 1$. This is because nonterminals with fan-out one can always be treated as in the context-free grammar case, as it will be explained later.

Example 3 The p-graph associated with the LCFRS production in Example 2 is shown in Figure 2. Circled sets of edges indicate the factorization in that example. □

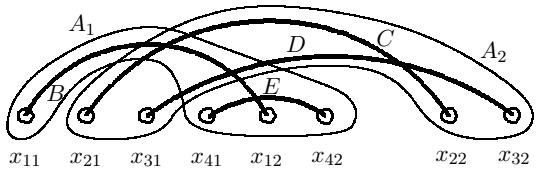

Figure 2: The p-graph associated with the LCFRS production in Example 2.

We close this section by introducing some additional notation related to p-graphs that will be used throughout this paper. Let $E \subseteq E_p$ be some set of edges. The **cover** set for E is defined as $V(E) = \{x \mid (x, x') \in E\}$ (recall that our edges are unordered pairs, so (x, x') and (x', x) denote the same edge). Conversely, let $V \subseteq V_p$ be some set of vertices. The **incident** set for V is defined as $E(V) = \{(x, x') \mid (x, x') \in E_p,\ x \in V\}$.

Assume $\varphi(p) = 2$, and let $x_1, x_2 \in V_p$. If x_1 and x_2 do not occur both in the same string y_1 or y_2, then we say that there is a **gap** between x_1 and x_2. If $x_1 \prec_p x_2$ and there is no gap between x_1 and x_2, then we write $[x_1, x_2]$ to denote the set $\{x_1, x_2\} \cup \{x \mid x \in V_p,\ x_1 \prec_p x \prec_p x_2\}$. For $x \in V_p$ we also let $[x, x] = \{x\}$. A set $[x, x']$ is called a **range**. Let r and r' be two ranges. The pair (r, r') is called a **tandem** if the following conditions are both satisfied: (i) $r \cup r'$ is not a range, and (ii) there exists some edge $(x, x') \in E_p$ with $x \in r$ and $x' \in r'$. Note that the first condition means that r and r' are disjoint sets and, for any pair of vertices $x \in r$ and $x' \in r'$, either there is a gap between x and x' or else there exists some $x_g \in V_p$ such that $x \prec_p x_g \prec_p x'$ and $x_g \notin r \cup r'$.

A set of edges $E \subseteq E_p$ is called a **bundle** with fan-out one if $V(E) = [x_1, x_2]$ for some $x_1, x_2 \in V_p$, i.e., $V(E)$ is a range. Set E is called a bundle with fan-out two if $V(E) = [x_1, x_2] \cup [x_3, x_4]$ for some $x_1, x_2, x_3, x_4 \in V_p$, and $([x_1, x_2], [x_3, x_4])$ is a tandem. Note that if E is a bundle with fan-out two with $V(E) = [x_1, x_2] \cup [x_3, x_4]$, then neither $E([x_1, x_2])$ nor $E([x_3, x_4])$ are bundles with fan-out one, since there is at least one edge incident upon a vertex in $[x_1, x_2]$ and a vertex in $[x_3, x_4]$. We also use the term bundle to denote a bundle with fan-out either one or two.

Intuitively, in a p-graph associated with a LCFRS production p, a bundle E with fan-out f and with $|E| > 1$ identifies a set of nonterminals $\mathcal{C}$ in the right-hand side of p that can be factorized into a new production. The nonterminals in $\mathcal{C}$ are then replaced in p by a fresh nonterminal C with fan-out f, as already explained. Our factorization algorithm is based on efficient methods for the detection of bundles with fan-out one and two.

4 The algorithm

In this section we provide an efficient, recursive algorithm for the decomposition of a p-graph into bundles, which corresponds to factorizing the represented LCFRS production.

4.1 Overview of the algorithm

The basic idea underlying our graph-based algorithm can be described as follows. We want to compute an optimal hierarchical decomposition of an input bundle with fan-out 1 or 2. This decomposition can be represented by a tree, in which each node N corresponds to a bundle (the root node corresponds to the input bundle) and the daughters of N represent the bundles in which N is immediately decomposed. The decomposition is optimal in so far as the maximum arity of the decomposition tree is as small as possible. As already explained above, this decomposition represents a factorization of some production p of a LCFRS, resulting in optimal rank reduction. All the internal nodes in the decomposition represent fresh nonterminals that will be created during the factorization process.

The construction of the decomposition tree is carried out recursively. For a given bundle with fan-out 1 or 2, we apply a procedure for decomposing this bundle in its immediate sub-bundles with fan-out 1 or 2, in an optimal way. Then,

we recursively apply our procedure to the obtained sub-bundles. Recursion stops when we reach bundles containing only one edge (which correspond to the nonterminals in the right-hand side of the input production). We shall prove that the result is an optimal decomposition.

The procedure for computing an optimal decomposition of a bundle F into its immediate sub-bundles, which we describe in the first part of this section, can be sketched as follows. First, we identify and temporarily remove all maximal bundles with fan-out 1 (Section 4.3). The result is a new bundle F' which is a subset of the original bundle, and has the same fan-out. Next, we identify all sub-bundles with fan-out 2 in F' (Section 4.4). We compute the optimal decomposition of F', resting on the hypothesis that there are no sub-bundles with fan-out 1. Each resulting sub-bundle is later expanded with the maximal sub-bundles with fan-out 1 that have been previously removed. This results in a "first level" decomposition of the original bundle F. We then recursively decompose all individual sub-bundles of F, including the bundles with fan-out 1 that have been later attached.

4.2 Backward and forward quantities

For a set $V \subseteq V_p$ of vertices, we write $\max(V)$ (resp. $\min(V)$) the maximum (resp. minimum) vertex in V w.r.t. the $\prec_p$ total order.

Let $r = [x_1, x_2]$ be a range. We write $r.left = x_1$ and $r.right = x_2$. The set of backward edges for r is defined as $B_r = \{(x, x') \,|\, (x, x') \in E_r,\ x \prec_p r.left,\ x' \in r\}$. The set of forward edges for r is defined symmetrically as $F_r = \{(x, x') \,|\, (x, x') \in E_r,\ x \in r,\ r.right \prec_p x'\}$. For $E \in \{B_r, F_r\}$ we also define $L(E) = \{x \,|\, (x, x') \in E,\ x \prec_p x'\}$ and $R(E) = \{x' \,|\, (x, x') \in E,\ x \prec_p x'\}$.

Let us assume $B_r \neq \emptyset$. We write $r.b.left = \min(L(B_r))$. Intuitively, $r.b.left$ is the leftmost vertex of the p-graph that is located at the left of range r and that is connected to some vertex in r through some edge. Similarly, we write $r.b.right = \max(L(B_r))$. If $B_r = \emptyset$, then we set $r.b.left = r.b.right = \bot$. Quantities $r.b.left$ and $r.b.right$ are called **backward** quantities.

We also introduce **local backward** quantities, defined as follows. We write $r.lb.left = \min(R(B_r))$. Intuitively, $r.lb.left$ is the leftmost vertex among all those vertices in r that are connected to some vertex to the left of r. Similarly, we write $r.lb.right = \max(R(B_r))$. If $B_r = \emptyset$, then we set $r.lb.left = r.lb.right = \bot$.

We define **forward** and **local forward** quantities in a symmetrical way.

The backward quantities $r.b.left$ and $r.b.right$ and the local backward quantities $r.lb.left$ and $r.lb.right$ for all ranges r in the p-graph can be computed efficiently as follows. We process ranges in increasing order of size, expanding each range r by one unit at a time by adding a new vertex at its right. Backward and local backward quantities for the expanded range can be expressed as a function of the same quantities for r. Therefore if we store our quantities for previously processed ranges, each new range can be annotated with the desired quantities in constant time. This algorithm runs in time $\mathcal{O}(n^2)$, where n is the number of vertices in V_p. This is an optimal result, since $\mathcal{O}(n^2)$ is also the size of the output.

We compute in a similar way the forward quantities $r.f.left$ and $r.f.right$ and the local forward quantities $r.lf.left$ and $r.lf.right$, this time expanding each range by one unit at its left.

4.3 Bundles with fan-out one

The detection of bundles with fan-out 1 within the p-graph can be easily performed in $\mathcal{O}(n^2)$, where n is the number of its vertices. Indeed, the incident set $E(r)$ of a range r is a bundle with fan-out one if and only if $r.b.left = r.f.left = \bot$. This immediately follows from the definitions given in Section 4.2. It is therefore possible to check all ranges the one after the other, once the backward and forward properties have been computed. These checks take constant time for each of the $\Theta(n^2)$ ranges, hence the quadratic complexity.

We now remove from F all bundles with fan-out 1 from the original bundle F. The result is the new bundle F', that has no sub-bundles with fan-out 1.

4.4 Bundles with fan-out two

Efficient detection of bundles with fan-out two in F' is considerably more challenging. A direct generalization of the technique proposed for detecting bundles with fan-out 1 would use the following property, that is also a direct corollary of the definitions in Section 4.2: the incident set $E(r \cup r')$ of a tandem (r, r') is a bundle with fan-out two if and only if all of the following conditions hold: (i) $r.b.left = r'.f.left = \bot$, (ii) $r.f.left \in r'$, $r.f.right \in r'$, (iii) $r'.b.left \in r$, $r'.b.right \in r$.

However, checking all $\mathcal{O}(n^4)$ tandems the one after the other would require time $\mathcal{O}(n^4)$. Therefore, preserving the quadratic complexity of the overall algorithm requires a more complex representation.

From now on, we assume that $V_p = \{x_1, \ldots, x_n\}$, and we write $[i, j]$ as a shorthand for the range $[x_i, x_j]$.

First, we need to compute an additional data structure that will store local backward figures in a convenient way. Let us define the **expansion table** T as follows: for a given range $r' = [i', j']$, $T(r')$ is the set of all ranges $r = [i, j]$ such that $r.lb.left = i'$ and $r.lb.right = j'$, ordered by increasing left boundary i. It turns out that the construction of such a table can be achieved in time $\mathcal{O}(n^2)$. Moreover, it is possible to compute in $\mathcal{O}(n^2)$ an auxiliary table T' that associates with r the first range r'' in $T([r.f.left, r.f.right])$ such that $r''.b.right \geq r$. Therefore, either $(r, T'(r))$ anchors a valid bundle, or there is no bundle E such that the first component of $V(E)$ is r.

We now have all the pieces to extract bundles with fan-out 2 in time $\mathcal{O}(n^2)$. We proceed as follows. For each range $r = [i, j]$:

- We first retrieve $r' = [r.f.left, r.f.right]$ in constant time.
- Then, we check in constant time whether $r'.b.left$ lies within r. If it doesn't, r is not the first part of a valid bundle with fan-out 2, and we move on to the next range r.
- Finally, for each r'' in the ordered set $T(r')$, starting with $T'(r)$, we check whether $r''.b.right$ is inside r. If it is not, we stop and move on to the next range r. If it is, we output the valid bundle (r, r'') and move on to the next element in $T(r')$. Indeed, in case of a failure, the backward edge that relates a vertex in r'' with a vertex outside r will still be included in all further elements in $T(r')$ since $T(r')$ is ordered by increasing left boundary. This step costs a constant time for each success, and a constant time for the unique failure, if any.

This algorithm spends a constant time on each range plus a constant time on each bundle with fan-out 2. We shall prove in Section 5 that there are $\mathcal{O}(n^2)$ bundles with fan-out 2. Therefore, this algorithm runs in time $\mathcal{O}(n^2)$.

Now that we have extracted all bundles, we need to extract an optimal decomposition of the input bundle F', i.e., a minimal size partition of all n elements (edges) in the input bundle such that each of these partition is a bundle (with fan-out 2, since bundles with fan-out 1 are excluded, except for the input bundle). By definition, a partition has minimal size if there is no other partition it is a refinment of.[3]

4.5 Extracting an optimal decomposition

We have constructed the set of all (fan-out 2) sub-bundles of F'. We now need to build one optimal decomposition of F' into sub-bundles. We need some more theoretical results on the properties of bundles.

Lemma 1 *Let E_1 and E_2 be two sub-bundles of F' (with fan-out 2) that have non-empty intersection, but that are not included the one in the other. Then $E_1 \cup E_2$ is a bundle (with fan-out 2).*

PROOF This lemma can be proved by considering all possible respective positions of the covers of E_1 and E_2, and discarding all situations that would lead to the existence of a fan-out 1 sub-bundle. ■

Theorem 1 *For any bundle E, either it has at least one binary decomposition, or all its decompositions are refinements of a unique optimal one.*

PROOF Let us suppose that E has no binary decomposition. Its cover corresponds to the tandem $(r, r') = ([i, j], [i', j'])$. Let us consider two different decompositions of E, that correspond respectively to decompositions of the range r in two sets of sub-ranges of the form $[i, k_1], [k_1 + 1, k_2], \ldots, [k_m, j]$ and $[i, k'_1], [k'_1 + 1, k'_2], \ldots, [k'_{m'}, j]$. For simplifying the notations, we write $k_0 = k'_0 = i$ and $k_{m+1} = k_{m'+1} = j$. Since $k_0 = k'_0$, there exist an index $p > 0$ such that for any $l < p$, $k_l = k'_l$, but $k_p \neq k'_p$: p is the index that identifies the first discrepancy between both decomposition. Since $k_{m+1} = k_{m'+1}$, there must exist $q \leq m$ and $q' \leq m'$ such that q and q' are strictly greater than p and that are the minimal indexes such that $k_q = k'_{q'}$. By definition, all bundles of the form $E_{[k_{l-1}, k_l]}$ ($p \leq l \leq q$) have a non-empty intersection with at least one bundle of the form $E_{[k'_{l-1}, k'_l]}$

[3] The term "refinement" is used in the usual way concerning partitions, i.e., a partition P_1 is a refinement of another one P_2 if all constituents in P_1 are constituents of P_2, or belongs to a subset of the partition P_1 that is a partition of one element of P_2.

($p \leq l \leq q'$). The reverse is true as well. Applying Lemma 1, this shows that $E([k_{p+1}, k_q])$ is a bundle with fan-out 2. Therefore, by replacing all ranges involved in this union in one decomposition or the other, we get a third decomposition for which the two initial ones are strict refinements. This is a contradiction, which concludes the proof. ∎

Lemma 2 *Let $E = V(r \cup r')$ be a bundle, with $r = [i, j]$. We suppose it has a unique (non-binary) optimal decomposition, which decomposes $[i, j]$ into $[i, k_1], [k_1 + 1, k_2], \ldots, [k_m, j]$. There exist no range $r'' \subset r$ such that (i) $E_{r''}$ is a bundle and (ii) $\exists l$, $1 \leq l \leq m$ such that $[k_l, k_{l+1}] \subset r''$.*

PROOF Let us consider a range r'' that would contradict the lemma. The union of r'' and of the ranges in the optimal decomposition that have a non-empty intersection with r'' is a fan-out 2 bundle that includes at least two elements of the optimal decomposition, but that is strictly included in E because the decomposition is not binary. This is a contradiction. ∎

Lemma 3 *Let $E = V(r, r')$ be a bundle, with $r = [i, j]$. We suppose it has a binary (optimal) decomposition (not necessarily unique). Let $r'' = [i, k]$ be the largest range starting in i such that $k < j$ and such that it anchors a bundle, namely $E(r'')$. Then $E(r'')$ and $E([k + 1, j])$ form a binary decomposition of E.*

PROOF We need to prove that $E([k + 1, j])$ is a bundle. Each (optimal) binary decomposition of E decomposes r in 1, 2 or 3 sub-ranges. If no optimal decomposition decomposes r in at least 2 sub-ranges, then the proof given here can be adapted by reasoning on r' instead of r. We now suppose that at least one of them decomposes r in at least 2 sub-ranges. Therefore, it decomposes r in $[i, k_1]$ and $[k_1 + 1, j]$ or in $[i, k_1]$, $[k_1 + 1, k_2]$ and $[k_2 + 1, j]$. We select one of these optimal decomposition by taking one such that k_1 is maximal. We shall now distinguish between two cases.

First, let us suppose that r is decomposed into two sub-ranges $[i, k_1]$ and $[k_1 + 1, j]$ by the selected optimal decomposition. Obviously, $E([i, k_1])$ is a "crossing" bundle, i.e., the right component of its cover is is a sub-range of r'. Since r is decomposed in two sub-ranges, it is necessarily the same for r'. Therefore, $E([i, k_1])$ has a cover of the form $[i, k_1] \cup [i', k'_1]$ or $[i, k_1] \cup [k'_1 + 1, j]$. Since r'' includes $[i, k_1]$, $E(r'')$ has a cover of the form $[i, k] \cup [i', k']$ or $[i, k] \cup [k' + 1, j]$. This means that r' is decomposed by $E(r'')$ in only 2 ranges, namely the right component of $E(r'')$'s cover and another range, that we can call r'''. Since $r \setminus r'' = [k + 1, j]$ may not anchor a bundle with fan-out 1, it must contain at least one crossing edge. All such edges necessarily fall within r'''. Conversely, any crossing edge that falls inside r''' necessarily has its other end inside $[k + 1, j]$. Which means that $E(r'')$ and $E(r''')$ form a binary decomposition of E. Therefore, by definition of k_1, $k = k_1$.

Second, let us suppose that r is decomposed into 3 sub-ranges by the selected original decomposition (therefore, r' is not decomposed by this decomposition). This means that this decomposition involves a bundle with a cover of the form $[i, k_1] \cup [k_2 + 1, j]$ and another bundle with a cover of the form $[k_1 + 1, k_2] \cup r'$ (this bundle is in fact $E(r')$). If $k \geq k_2$, then the left range of both members of the original decomposition are included in r'', which means that $E(r'') = E$, and therefore $r'' = r$ which is excluded. Note that k is at least as large as k_1 (since $[i, k_1]$ is a valid "range starting in i such that $k < j$ and such that it anchors a bundle"). Therefore, we have $k_1 \leq k < k_2$. Therefore, $E([i, k_1]) \subset E(r'')$, which means that all edges anchored inside $[k_2 + 1, j])$ are included in $E(r'')$. Hence, $E(r'')$ can not be a crossing bundle without having a left component that is $[i, j]$, which is excluded (it would mean $E(r'') = E$). This means that $E(r'')$ is a bundle with a cover of the form $[i, k] \cup [k' + 1, j]$. Which means that $E(r')$ is in fact the bundle whose cover is $[k + 1, k' + 1] \cup r'$. Hence, $E(r'')$ and $E(r')$ form a binary decomposition of E. Hence, by definition of k_1, $k = k_1$. ∎

As an immediate consequence of Lemmas 2 and 3, our algorithm for extracting the optimal decomposition for F' consists in applying the following procedure recursively, starting with F', and repeating it on each constructed sub-bundle E, until sub-bundles with only one edge are reached.

Let $E = E(r, r')$ be a bundle, with $r = [i, j]$. One optimal decomposition of E can be obtained as follows. One selects the bundle with a left component starting in i and with the maximum length, and iterating this selection process until r is covered. The same is done with r'. We retain the optimal among both resulting decompositions (or one of them if they are both optimal). Note that this

decomposition is unique if and only if it has four components or more; it can not be ternary; it may be binary, and in this case it may be non-unique.

This algorithm gives us a way to extract an optimal decomposition of F' in linear time w.r.t. the number of sub-bundles in this optimal decomposition. The only required data structure is, for each i (resp. k), the list of bundles with a cover of the form $[i,j] \cup [k,l]$ ordered by decreasing j (resp. l). This can trivially be constructed in time $\mathcal{O}(n^2)$ from the list of all bundles we built in time $\mathcal{O}(n^2)$ in the previous section. Since the number of bundles is bounded by $\mathcal{O}(n^2)$ (as mentioned above and proved in Section 5), this means we can extract an optimal decomposition for F' in $\mathcal{O}(n^2)$.

Similar ideas apply to the simpler case of the decomposition of bundles with fan-out 1.

4.6 The main decomposition algorithm

We now have to generalize our algorithm in order to handle the possible existence of fan-out 1 bundles. We achieve this by using the fan-out 2 algorithm recursively. First, we extract and remove (maximal) bundles with fan-out 1 from F, and recursively apply to each of them the complete algorithm. What remains is F', which is a set of bundles with no sub-bundles with fan-out 1. This means we can apply the algorithm presented above. Then, for each bundle with fan-out 1, we group it with a randomly chosen adjacent bundle with fan-out 2, which builds an expanded bundle with fan-out 2, which has a binary decomposition into the original bundle with fan-out 2 and the bundle with fan-out 1.

5 Time complexity analysis

In Section 4, we claimed that there are no more than $\mathcal{O}(n^2)$ bundles. In this section we sketch the proof of this result, which will prove the quadratic time complexity of our algorithm.

Let us compute an upper bound on the number of bundles with fan-out two that can be found within the p-graph processed in Section 4.5, i.e., a p-graph with no fan-out 1 sub-bundle.

Let $E, E' \subseteq E_p$ be bundles with fan-out two. If $E \subset E'$, then we say that E' **expands** E. E' is said to **immediately** expand E, written $E \rightarrow E'$, if E' expands E and there is no bundle E'' such that E'' expands E and E' expands E''.

Let us represent bundles and the associated immediate expansion relation by means of a graph. Let $\mathcal{E}$ denote the set of all bundles (with fan-out two) in our p-graph. The **e-graph** associated with our LCFRS production p is the directed graph with vertices $\mathcal{E}$ and edges defined by the relation $\rightarrow$. For $E \in \mathcal{E}$, we let $out(E) = \{E' \mid E \rightarrow E'\}$ and $in(E) = \{E' \mid E' \rightarrow E\}$.

Lack of space prevents us from providing the proof of the following property. For any $E \in \mathcal{E}$ that contains more than one edge, $|out(E)| \leq 2$ and $|in(E)| \geq 2$. This allows us to prove our upper bound on the size of $\mathcal{E}$.

Theorem 2 *The e-graph associated with an* LCFRS *production p has at most n^2 vertices, where n is the rank of p.*

PROOF Consider the e-graph associated with production p, with set of vertices $\mathcal{E}$. For a vertex $E \in \mathcal{E}$, we define the **level** of E as the number $|E|$ of edges in the corresponding bundle from the p-graph associated with p. Let d be the maximum level of a vertex in $\mathcal{E}$. We thus have $1 \leq d \leq n$. We now prove the following claim. For any integer k with $1 \leq k \leq d$, the set of vertices in $\mathcal{E}$ with level k has no more than n elements.

For $k = 1$, since there are no more than n edges in such a p-graph, the statement holds.

We can now consider all vertices in $\mathcal{E}$ with level $k > 1$ ($k \leq d$). Let $\mathcal{E}^{(k-1)}$ be the set of all vertices in $\mathcal{E}$ with level smaller than or equal to $k-1$, and let us call $T^{(k-1)}$ the set of all edges in the e-graph that are leaving from some vertex in $\mathcal{E}^{(k-1)}$. Since for each bundle E in $\mathcal{E}^{(k-1)}$ we know that $|out(E)| \leq 2$, we have $|T^{(k-1)}| \leq 2|\mathcal{E}^{(k-1)}|$.

The number of vertices in $\mathcal{E}^{(k)}$ with level larger than one is at least $|\mathcal{E}^{(k-1)}| - n$. Since for each $E \in \mathcal{E}^{(k-1)}$ we know that $|in(E)| \geq 2$, we conclude that at least $2(|\mathcal{E}^{(k-1)}| - n)$ edges in $T^{(k-1)}$ must end up at some vertex in $\mathcal{E}^{(k)}$. Let T be the set of edges in $T^{(k-1)}$ that impinge on some vertex in $\mathcal{E} \setminus \mathcal{E}^{(k)}$. Thus we have $|T| \leq 2|\mathcal{E}^{(k-1)}| - 2(|\mathcal{E}^{(k-1)}| - n) = 2n$. Since the vertices of level k in $\mathcal{E}$ must have incoming edges from set T, and because each of them have at least 2 incoming edges, there cannot be more than n such vertices. This concludes the proof of our claim.

Since the the level of a vertex in $\mathcal{E}$ is necessarily lower than n, this completes the proof. ∎

The overall complexity of the complete algorithm can be computed by induction. Our induction hypothesis is that for $m < n$, the time complexity is in $\mathcal{O}(m^2)$. This is obviously true for $n = 1$ and $n = 2$. Extracting the bundles

with fan-out 1 costs $\mathcal{O}(n^2)$. These bundles are of length $n_1 \dots n_m$. Extracting bundles with fan-out 2 costs $\mathcal{O}((n - n_1 - \dots - n_m)^2)$. Applying recursively the algorithm to bundles with fan-out 1 costs $\mathcal{O}(n_1^2) + \dots + \mathcal{O}(n_m^2)$. Therefore, the complexity is in $\mathcal{O}(n^2) + \mathcal{O}((n - n_1 - \dots - n_m)^2) + \sum_{i=1}^{n} \mathcal{O}(n_i) = \mathcal{O}(n^2) + \mathcal{O}(\sum_{i=1}^{n} n_i) = \mathcal{O}(n^2)$.

6 Conclusion

We have introduced an efficient algorithm for optimal reduction of the rank of LCFRSs with fan-out at most 2, that runs in quadratic time w.r.t. the rank of the input grammar. Given the fact that fan-out 1 bundles can be attached to any adjacent bundle in our factorization, we can show that our algorithm also optimizes time complexity for known tabular parsing algorithms for LCFRSs with fan-out 2.

As for general LCFRS, it has been shown by Gildea (2010) that rank optimization and time complexity optimization are not equivalent. Furthermore, all known algorithms for rank or time complexity optimization have an exponential time complexity (Gómez-Rodríguez et al., 2009).

Acknowledgments

Part of this work was done while the second author was a visiting scientist at Alpage (INRIA Paris-Rocquencourt and Université Paris 7), and was financially supported by the hosting institutions.

References

Daniel Gildea. 2010. Optimal parsing strategies for linear context-free rewriting systems. In *Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, Los Angeles, California. To appear.

Carlos Gómez-Rodríguez and Giorgio Satta. 2009. An optimal-time binarization algorithm for linear context-free rewriting systems with fan-out two. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 985–993, Suntec, Singapore, August. Association for Computational Linguistics.

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

Michael A. Harrison. 1978. *Introduction to Formal Language Theory*. Addison-Wesley, Reading, MA.

Aravind K. Joshi and Leon S. Levy. 1977. Constraints on local descriptions: Local transformations. *SIAM Journal of Computing*

Marco Kuhlmann and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proceedings of the 12th Meeting of the European Chapter of the Association for Computational Linguistics (EACL 2009)*, Athens, Greece. To appear.

Wolfgang Maier and Timm Lichte. 2009. Characterizing discontinuity in constituent treebanks. In *Proceedings of the 14th Conference on Formal Grammar (FG 2009)*, Bordeaux, France.

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

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

Giorgio Satta. 1992. Recognition of linear context-free rewriting systems. In *Proceedings of the 30th Meeting of the Association for Computational Linguistics (ACL'92)*, Newark, Delaware.

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)*.

David J. Weir. 1992. Linear context-free rewriting systems and deterministic tree-walk transducers. In *Proceedings of the 30th Meeting of the Association for Computational Linguistics (ACL'92)*, Newark, Delaware.