Optimal Reduction of Rule Length in Linear Context-Free Rewriting Systems

Carlos Gómez-Rodríguez[1], **Marco Kuhlmann**[2], **Giorgio Satta**[3] **and David Weir**[4]

[1] Departamento de Computación, Universidade da Coruña, Spain (`cgomezr@udc.es`)

[2] Department of Linguistics and Philology, Uppsala University, Sweden (`marco.kuhlmann@lingfil.uu.se`)

[3] Department of Information Engineering, University of Padua, Italy (`satta@dei.unipd.it`)

[4] Department of Informatics, University of Sussex, United Kingdom (`davidw@sussex.ac.uk`)

Abstract

Linear Context-free Rewriting Systems (LCFRS) is an expressive grammar formalism with applications in syntax-based machine translation. The parsing complexity of an LCFRS is exponential in both the *rank* of a production, defined as the number of nonterminals on its right-hand side, and a measure for the discontinuity of a phrase, called *fan-out*. In this paper, we present an algorithm that transforms an LCFRS into a strongly equivalent form in which all productions have rank at most 2, and has minimal fan-out. Our results generalize previous work on Synchronous Context-Free Grammar, and are particularly relevant for machine translation from or to languages that require syntactic analyses with discontinuous constituents.

1 Introduction

There is currently considerable interest in syntax-based models for statistical machine translation that are based on the extraction of a synchronous grammar from a corpus of word-aligned parallel texts; see for instance Chiang (2007) and the references therein. One practical problem with this approach, apart from the sheer number of the rules that result from the extraction procedure, is that the parsing complexity of all synchronous formalisms that we are aware of is exponential in the **rank** of a rule, defined as the number of nonterminals on the right-hand side. Therefore, it is important that the rules of the extracted grammar are transformed so as to minimise this quantity. Not only is this beneficial in terms of parsing complexity, but smaller rules can also improve a translation model's ability to generalize to new data (Zhang et al., 2006).

Optimal algorithms exist for minimising the size of rules in a Synchronous Context-Free Grammar (SCFG) (Uno and Yagiura, 2000; Zhang et al., 2008). However, the SCFG formalism is limited to modelling word-to-word alignments in which a single continuous phrase in the source language is aligned with a single continuous phrase in the target language; as defined below, this amounts to saying that SCFG have a **fan-out** of 2. This restriction appears to render SCFG empirically inadequate. In particular, Wellington et al. (2006) find that the coverage of a translation model can increase dramatically when one allows a bilingual phrase to stretch out over three rather than two continuous substrings. This observation is in line with empirical studies in the context of dependency parsing, where the need for formalisms with higher fan-out has been observed even in standard, single language texts (Kuhlmann and Nivre, 2006).

In this paper, we present an algorithm that computes optimal decompositions of rules in the formalism of Linear Context-Free Rewriting Systems (LCFRS) (Vijay-Shanker et al., 1987). LCFRS was originally introduced as a generalization of several so-called mildly context-sensitive grammar formalisms. In the context of machine translation, LCFRS is an interesting generalization of SCFG because it does not restrict the fan-out to 2, allowing productions with *arbitrary* fan-out (and arbitrary rank). Given an LCFRS, our algorithm computes a strongly equivalent grammar with rank 2 and minimal increase in fan-out.[1] In this context, strong equivalence means that the derivations of the original grammar can be reconstructed using some simple homomorphism (c.f. Nijholt, 1980). Our contribution is significant because the existing algorithms for decomposing SCFG, based on Uno and Yagiura (2000), cannot be applied to LCFRS, as they rely on the crucial property that components of biphrases are strictly separated in the generated string: Given a pair of synchronized nonterminal symbols, the material derived from the source nonterminal must precede the material derived from the target nonterminal, or vice versa. The problem that we solve has been previously addressed by Melamed et al. (2004), but in contrast to our result, their algorithm does not guarantee an optimal (minimal) increase in the fan-out of the resulting grammar. However, this is essential for the practical applicability of the transformed grammar, as the parsing complexity of LCFRS is exponential in *both* the rank and the fan-out.

Structure of the paper The remainder of the paper is structured as follows. Section 2 introduces the terminology and notation that we use for LCFRS. In Section 3, we present the technical background of our algorithm; the algorithm itself is discussed in Section 4. Section 5 concludes the paper by discussing related work and open problems.

General notation The set of non-negative integers is denoted by $\mathbb{N}$. For $i, j \in \mathbb{N}$, we write $[i, j]$ to denote the interval $\{ k \in \mathbb{N} \mid i \leq k \leq j \}$, and use $[i]$ as a shorthand for $[1, i]$. Given an alphabet V, we write V^* for the set of all (finite) strings over V.

2 Preliminaries

We briefly summarize the terminology and notation that we adopt for LCFRS; for detailed definitions, see Vijay-Shanker et al. (1987).

2.1 Linear, non-erasing functions

Let V be an alphabet. For natural numbers $r \geq 0$ and $f, f_1, \dots, f_r \geq 1$, a function

$$g : (V^*)^{f_1} \times \cdots \times (V^*)^{f_r} \to (V^*)^f$$

[1]Rambow and Satta (1999) show that without increasing fan-out it is not always possible to produce even *weakly* equivalent grammars.

is called a **linear, non-erasing function** over V of type $f_1 \times \cdots \times f_r \to f$, if it can be defined by an equation of the form

$$g(\langle x_{1,1}, \dots, x_{1,f_1} \rangle, \dots, \langle x_{r,1}, \dots, x_{r,f_r} \rangle) = \beta_g ,$$

where $\beta_g = \langle \alpha_{g,1}, \dots, \alpha_{g,f} \rangle$ is an f-tuple of strings over the variables on the left-hand side of the equation and symbols in V that contains exactly one occurrence of each variable. We call the value r the **rank** of g, the value f its **fan-out**, and write $\rho(g)$ and $\varphi(g)$, respectively, to denote these quantities. Note that, if we assume the variables on the left-hand side of the defining equation of g to be named according to the specific schema given above, then g is uniquely determined by β_g.

2.2 Linear context-free rewriting systems

A **linear context-free rewriting system** (LCFRS) is a construct $G = (V_N, V_T, P, S)$, where: V_N is an alphabet of nonterminal symbols in which each symbol $A \in V_N$ is associated with a value $\varphi(A)$, called its **fan-out**; V_T is an alphabet of terminal symbols; $S \in N$ is a distinguished start symbol with $\varphi(S) = 1$; and P is a set of productions of the form

$$p : A \to g(B_1, B_2, \dots, B_r) ,$$

where $A, B_1, \dots, B_r \in V_N$, and g is a linear, non-erasing function over the terminal alphabet V_T of type $\varphi(B_1) \times \cdots \times \varphi(B_r) \to \varphi(A)$. In a derivation of an LCFRS, the production p can be used to transform a sequence of r tuples of strings, generated by the nonterminals $B_1, \dots, B_r$, into a single $\varphi(A)$-tuple of strings, associated with the nonterminal A. The values $\rho(g)$ and $\varphi(g)$ are called the **rank** and **fan-out** of p, respectively, and we write $\rho(p)$ and $\varphi(p)$, respectively, to denote these quantities. The rank and fan-out of G, written $\rho(G)$ and $\varphi(G)$, respectively, are the maximum rank and fan-out among all of its productions. Given that $\varphi(S) = 1$, a derivation will associate S with a set of one-component tuples of strings over V_T; this forms the string language generated by G.

Example 1 The following LCFRS generates the string language $\{ a^n b^n c^n d^n \mid n \in \mathbb{N} \}$. We only specify the set of productions; the remaining components of the grammar are obvious from that.

$$\begin{array}{ll} S \to g_1(R) & g_1(\langle x_{1,1}, x_{1,2}\rangle) = \langle x_{1,1}x_{1,2}\rangle \\ R \to g_2(R) & g_2(\langle x_{1,1}, x_{1,2}\rangle) = \langle ax_{1,1}b, cx_{1,2}d\rangle \\ R \to g_3 & g_3 = \langle \varepsilon, \varepsilon\rangle \end{array}$$

The functions g_1 and g_2 have rank 1; the function g_3 has rank 0. The functions g_2 and g_3 have fan-out 2; the function g_1 has fan-out 1. □

3 Technical background

The general idea behind our algorithm is to replace each production of an LCFRS with a set of "shorter" productions that jointly are equivalent to the original production. Before formalizing this idea, we first introduce a specialized representation for the productions of an LCFRS.

We distinguish between occurrences of symbols within a string by exploiting two different notations. Let $\alpha = a_1a_2 \cdots a_n$ be a string. The occurrence a_i in α can be denoted by means of its **position** index $i \in [n]$, or else by means of its two (left and right) **endpoints**, $i-1$ and i; here, the left (right) endpoint denotes a boundary between occurrence a_i and the previous (subsequent) occurrence, or the beginning (end) of the string α. Similarly, a substring $a_i \cdots a_j$ of α with $i \leq j$ can be denoted by the positions $i, i+1, \ldots, j$ of its occurrences, or else by means of its left and right endpoints, $i-1$ and j.

3.1 Production representation

For the remainder of this section, let us fix an LCFRS $G = (V_N, V_T, P, S)$ and a production $p : A \to g(B_1, \ldots, B_r)$ of G, with g defined as in Section 2.1. We define

$$|p| = \varphi(g) + \sum_{i=1}^{\varphi(g)} |\alpha_{g,i}|.$$

Let \$ be a fresh symbol that does not occur in G. We define the **characteristic string** of the production p as

$$\sigma(p) = \alpha_{g,1}\$ \cdots \$\alpha_{g,\varphi(g)} ,$$

and the **variable string** of p as the string $\sigma_N(p)$ obtained from $\sigma(p)$ by removing all the occurrences of symbols in V_T.

Example 2 We will illustrate the concepts introduced in this section using the concrete production $p_0 : A \to g(B_1, B_2, B_3)$, where

$$\beta_g = \langle x_{1,1}ax_{2,1}x_{1,2}, x_{3,1}bx_{3,2}\rangle .$$

In this case, we have

$$\begin{aligned} \sigma(p_0) &= x_{1,1}ax_{2,1}x_{1,2}\$x_{3,1}bx_{3,2} , \quad \text{and} \\ \sigma_N(p_0) &= x_{1,1}x_{2,1}x_{1,2}\$x_{3,1}x_{3,2} . \end{aligned}$$ □

Let I be an index set, $I \subseteq [r]$. Consider the set $\mathcal{B}$ of occurrences B_i in the right-hand side of p such that $i \in I$.[2] We define the **position set** of $\mathcal{B}$, denoted by $\Pi_{\mathcal{B}}$, as the set of all positions $1 \leq j \leq |\sigma_N(p)|$ such that the jth symbol in $\sigma_N(p)$ is a variable of the form $x_{i,h}$, for $i \in I$ and some $h \geq 1$.

Example 3 Some position sets of p_0 are

$$\Pi_{\{B_1\}} = \{1, 3\} , \Pi_{\{B_2\}} = \{2\} , \Pi_{\{B_3\}} = \{5, 6\} .$$ □

A position set $\Pi_{\mathcal{B}}$ can be uniquely expressed as the union of $f \geq 1$ intervals $[l_1+1, r_1], \ldots, [l_f+1, r_f]$ such that $r_{i-1} < l_i$ for every $1 < i \leq f$. Thus we define the set of **endpoints** of $\Pi_{\mathcal{B}}$ as

$$\Delta_{\mathcal{B}} = \{ l_j \mid j \in [f] \} \cup \{ r_j \mid j \in [f] \} .$$

The quantity f is called the **fan-out** of $\Pi_{\mathcal{B}}$, written $\varphi(\Pi_{\mathcal{B}})$. Notice that the fan-out of a position set $\Pi_{\{B\}}$ does not necessarily coincide with the fan-out of the non-terminal B in the underlying LCFRS. A set with $2f$ endpoints always corresponds to a position set of fan-out f.

Example 4 For our running example, we have $\Delta_{\{B_1\}} = \{0, 1, 2, 3\}$, $\Delta_{\{B_2\}} = \{1, 2\}$, $\Delta_{\{B_3\}} = \{4, 6\}$. Consequently, the fan-out of $\Delta_{\{B_1\}}$ is 2, and the fan-out of $\Delta_{\{B_2\}}$ and $\Delta_{\{B_3\}}$ is 1. Notice that the fan-out of the non-terminal B_3 is 2. □

We drop $\mathcal{B}$ from $\Pi_{\mathcal{B}}$ and $\Delta_{\mathcal{B}}$ whenever this set is understood from the context or it is not relevant. Given a set of endpoints $\Delta = \{i_1, \ldots, i_{2f}\}$ with $i_1 < \cdots < i_{2f}$, we obtain its corresponding position set by calculating the **closure** of Δ, defined as

$$[\Delta] = \textstyle\bigcup_{j=1}^{f} [i_{2j-1}+1, i_{2j}] .$$

[2]To avoid clutter in our examples, we abuse the notation by not making an explicit distinction between nonterminals and occurrences of nonterminals in productions.

3.2 Reductions

Assume that $r > 2$. The **reduction** of p by the nonterminal occurrences B_{r-1}, B_r is the ordered pair of productions (p_1, p_2) that is defined as follows. Let $\gamma_1, \ldots, \gamma_n$ be the maximal substrings of $\sigma(p)$ that contain only variables $x_{i,j}$ with $r-1 \leq i \leq r$ and terminal symbols, and at least one variable. Then

$$p_1 : A \to g_1(B_1, \ldots, B_{r-2}, X) \qquad \text{and}$$
$$p_2 : X \to g_2(B_{r-1}, B_r)\,,$$

where X is a fresh nonterminal symbol, the characteristic string $\sigma(p_1)$ is the string obtained from $\sigma(p)$ by replacing each substring γ_i by the variable $x_{r-1,i}$, and the characteristic string $\sigma(p_2)$ is the string $\gamma_1\$\cdots\$\gamma_n$.

Note that the defining equations of neither g_1 nor g_2 are in the specific form discussed in Section 2.1; however, they can be brought into this form by a consistent renaming of the variables. We will silently assume this renaming to take place.

Example 5 The reduction of p_0 by the nonterminal occurrences B_2 and B_3 has $p_1 : A \to g_1(B_1, X)$ and $p_2 : X \to g_2(B_2, B_3)$ with

$$\sigma(p_1) = x_{1,1}x_{2,1}x_{1,2}\$x_{2,2}$$
$$\sigma(p_2) = ax_{2,1}\$x_{3,1}bx_{3,2}$$

or, after renaming and in standard notation,

$$g_1(\langle x_{1,1}, x_{1,2}\rangle, \langle x_{2,1}, x_{2,2}\rangle) = \langle x_{1,1}x_{2,1}x_{1,2}, x_{2,2}\rangle$$
$$g_2(\langle x_{1,1}\rangle, \langle x_{2,1}, x_{2,2}\rangle) = \langle ax_{1,1}, x_{2,1}bx_{2,2}\rangle\,.\square$$

It is easy to check that a reduction provides us with a pair of productions that are equivalent to the original production p, in terms of generative capacity, since

$$g_1(B_1, \ldots, B_{r-2}, g_2(B_{r-1}, B_r)) = g(B_1, \ldots, B_r)$$

for all tuples of strings generated from the nonterminals $B_1, \ldots, B_r$, respectively. Note also that the fan-out of production p_1 equals the fan-out of p. However, the fan-out of p_2 (the value n) may be greater than the fan-out of p, depending on the way variables are arranged in $\sigma(p)$. Thus, a reduction does not necessarily preserve the fan-out of the original production. In the worst case, the fan-out of p_2 can be as large as $\varphi(B_{r-1}) + \varphi(B_r)$.

```
1: Function NAIVE-BINARIZATION($p$)
2: result ← $\emptyset$;
3: currentProd ← $p$;
4: while $\rho$(currentProd) $> 2$ do
5:     $(p_1, p_2)$ ← any reduction of currentProd;
6:     result ← result $\cup$ $p_2$;
7:     currentProd ← $p_1$;
8: return result $\cup$ currentProd;
```

Figure 1: The naive algorithm

We have defined reductions only for the last two occurrences of nonterminals in the right-hand side of a production p. However, it is easy to see that we can also define the concept for two arbitrary (not necessarily adjacent) occurrences of nonterminals, at the cost of making the notation more complicated.

4 The algorithm

Let G be an LCFRS with $\varphi(G) = f$ and $\rho(G) = r$, and let $f' \geq f$ be a target fan-out. We will now present an algorithm that computes an equivalent LCFRS G' of fan-out at most f' whose rank is at most 2, if such an LCFRS exists in the first place. The algorithm works by exhaustively reducing all productions in G.

4.1 Naive algorithm

Given an LCFRS production p, a naive algorithm to compute an equivalent set of productions whose rank is at most 2 is given in Figure 1. By applying this algorithm to all the productions in the LCFRS G, we can obtain an equivalent LCFRS with rank 2. We will call such an LCFRS a **binarization** of G.

The fan-out of the obtained LCFRS will depend on the nonterminals that we choose for the reductions in line 5. It is not difficult to see that, in the worst case, the resulting fan-out can be as high as $\lceil \frac{r}{2} \rceil \cdot f$. This occurs when we choose $\lceil \frac{r}{2} \rceil$ nonterminals with fan-out f that have associated variables in the string $\sigma_N(p)$ that do not occur at consecutive positions.

The algorithm that we develop in Section 4.3 improves on the naive algorithm in that it can be exploited to find a sequence of reductions that results in a binarization of G that is optimal, i.e., leads to an LCFRS with *minimal* fan-out. The algorithm is based on a technical concept called *adjacency*.

4.2 Adjacency

Let p be some production in the LCFRS G, and let $\mathit{\Delta}_1, \mathit{\Delta}_2$ be sets of endpoints, associated with some sets of nonterminal occurrences in p. We say that $\mathit{\Delta}_1$ and $\mathit{\Delta}_2$ **overlap** if the intersection of their closures is nonempty, that is, if $[\mathit{\Delta}_1] \cap [\mathit{\Delta}_2] \neq \emptyset$. Overlapping holds if and only if the associated sets of nonterminal occurrences are not disjoint. If $\mathit{\Delta}_1$ and $\mathit{\Delta}_2$ do not overlap, we define their **merge** as

$$\oplus(\mathit{\Delta}_1, \mathit{\Delta}_2) \ = \ (\mathit{\Delta}_1 \cup \mathit{\Delta}_2) \setminus (\mathit{\Delta}_1 \cap \mathit{\Delta}_2) \,.$$

It is easy to see that $[\oplus(\mathit{\Delta}_1, \mathit{\Delta}_2)] = [\mathit{\Delta}_1] \cup [\mathit{\Delta}_2]$. We say that $\mathit{\Delta}_1$ and $\mathit{\Delta}_2$ are **adjacent** for a given fan-out f, written $\mathit{\Delta}_1 \leftrightarrow_f \mathit{\Delta}_2$, if $\mathit{\Delta}_1$ and $\mathit{\Delta}_2$ do not overlap, and $\varphi([\oplus(\mathit{\Delta}_1, \mathit{\Delta}_2)]) \leq f$.

Example 6 For the production p_0 from Example 2, we have $\oplus(\mathit{\Delta}_{\{B_1\}}, \mathit{\Delta}_{\{B_2\}}) = \{0, 3\}$, showing that $\mathit{\Delta}_{\{B_1\}} \leftrightarrow_1 \mathit{\Delta}_{\{B_2\}}$. Similarly, we have

$$\oplus(\mathit{\Delta}_{\{B_1\}}, \mathit{\Delta}_{\{B_3\}}) = \{0, 1, 2, 3, 4, 6\} \,,$$

showing that $\mathit{\Delta}_{\{B_1\}} \leftrightarrow_3 \mathit{\Delta}_{\{B_3\}}$, but that neither $\mathit{\Delta}_{\{B_1\}} \leftrightarrow_2 \mathit{\Delta}_{\{B_3\}}$ nor $\mathit{\Delta}_{\{B_1\}} \leftrightarrow_1 \mathit{\Delta}_{\{B_3\}}$ holds. □

4.3 Bounded binarization algorithm

The adjacency-based binarization algorithm is given in Figure 2. It starts with a working set containing the endpoint sets corresponding to each nonterminal occurrence in the input production p. Reductions of p are only explored for nonterminal occurrences whose endpoint sets are adjacent for the target fan-out f', since reductions not meeting this constraint would produce productions with fan-out greater than f'. Each reduction explored by the algorithm produces a new endpoint set, associated to the fresh nonterminal that it introduces, and this new endpoint set is added to the working set and potentially used in further reductions.

From the definition of the adjacency relation $\leftrightarrow_f$, it follows that at lines 9 and 10 of BOUNDED-BINARIZATION we only pick up reductions for p that do not exceed the fan-out bound of f'. This implies soundness for our algorithm. Completeness means that the algorithm fails only if there exists no binarization for p of fan-out not greater than f'. This property is intuitive if one observes that our algorithm is a specialization of standard algorithms for the computation of the closure of binary relations. A formal proof of this fact is rather long and tedious, and will not be reported here. We notice that there is a very close similarity between algorithm BOUNDED-BINARIZATION and the deduction procedure proposed by Shieber et al. (1995) for parsing. We discuss this more at length in Section 5.

```
1:  **Function** BOUNDED-BINARIZATION($p$, $f'$)
2:  workingSet ← $\emptyset$;
3:  agenda ← $\emptyset$;
4:  **for all** $i$ from 1 to $\rho(p)$ **do**
5:      workingSet ← workingSet $\cup \{\mathit{\Delta}_{\{B_i\}}\}$;
6:      agenda ← agenda $\cup \{\mathit{\Delta}_{\{B_i\}}\}$;
7:  **while** agenda $\neq \emptyset$ **do**
8:      $\mathit{\Delta}$ ← pop some endpoint set from agenda;
9:      **for all** $\mathit{\Delta}_1 \in$ workingSet with $\mathit{\Delta}_1 \leftrightarrow_{f'} \mathit{\Delta}$ **do**
10:         $\mathit{\Delta}_2 = \oplus(\mathit{\Delta}, \mathit{\Delta}_1)$;
11:         **if** $\mathit{\Delta}_2 \notin$ workingSet **then**
12:             workingSet ← workingSet $\cup \{\mathit{\Delta}_2\}$;
13:             agenda ← agenda $\cup \{\mathit{\Delta}_2\}$;
14: **if** $\mathit{\Delta}_{\{B_1, B_2, \ldots, B_{\rho(p))}\}} \in$ workingSet **then**
15:     **return true**;
16: **else**
17:     **return false**;
```

Figure 2: Algorithm to compute a bounded binarization

Note that we have expressed the algorithm as a decision function that will return true if there exists a binarization of p with fan-out not greater than f', and false otherwise. However, the algorithm can easily be modified to return a reduction producing such a binarization, by adding to each endpoint set $\mathit{\Delta} \in$ workingSet two pointers to the adjacent endpoint sets that were used to obtain it. If the algorithm is successful, the tree obtained by following these pointers from the final endpoint set $\mathit{\Delta}_{\{B_1, \ldots, B_{\rho(p)}\}} \in$ workingSet gives us a tree of reductions that will produce a binarization of p with fan-out not greater than f', where each node labeled with the set $\mathit{\Delta}_{\{B_i\}}$ corresponds to the nonterminal B_i, and nodes labeled with other endpoint sets correspond to the fresh nonterminals created by the reductions.

4.4 Implementation

In order to implement BOUNDED-BINARIZATION, we can represent endpoint sets in a canonical way as $2f'$-tuples of integer positions in ascending order, and with some special null value used to fill positions for endpoint sets with fan-out strictly smaller than f'. We will assume that the concrete null value is larger than any other integer.

We also need to provide some appropriate representation for the set workingSet, in order to guarantee efficient performance for the membership test and the insertion operation. Both operations can be implemented in constant time if we represent workingSet as an $(2 \times f')$-dimensional table with Boolean entries. Each dimension is indexed by values in $[0, n]$ plus our special null value; here n is the length of the string $\sigma_N(p)$, and thus $n = \mathcal{O}(|p|)$. However, this has the disadvantage of using space $\Theta(n^{2f'})$, even in case workingSet is sparse, and is affordable only for quite small values of f'. Alternatively, we can more compactly represent workingSet as a trie data structure. This representation has size certainly smaller than $2f' \times q$, where q is the size of the set workingSet. However, both membership and insertion operations take now an amount of time $\mathcal{O}(2f')$.

We now analyse the time complexity of algorithm BOUNDED-BINARIZATION for inputs p and f'. We first focus on the while-loop at lines 7 to 13. As already observed, the number of possible endpoint sets is bounded by $\mathcal{O}(n^{2f'})$. Furthermore, because of the test at line 11, no endpoint set is ever inserted into the agenda variable more than once in a single run of the algorithm. We then conclude that our while-loop cycles a number of times $\mathcal{O}(n^{2f'})$.

We now focus on the choice of the endpoint set Δ_1 in the inner for-loop at lines 9 to 13. Let us fix Δ as in line 8. It is not difficult to see that any Δ_1 with $\Delta_1 \leftrightarrow_{f'} \Delta$ must satisfy

$$\varphi(\Delta) + \varphi(\Delta_1) - |\Delta \cap \Delta_1| \quad \leq \quad f'. \qquad (1)$$

Let $I \subseteq \Delta$, and consider all endpoint sets Δ_1 with $\Delta \cap \Delta_1 = I$. Given (1), we also have

$$\varphi(\Delta_1) \quad \leq \quad f' + |I| - \varphi(\Delta). \qquad (2)$$

This means that, for each Δ coming out of the agenda, at line 9 we can choose all endpoint sets Δ_1 such that $\Delta_1 \leftrightarrow_{f'} \Delta$ by performing the following steps:

- arbitrarily choose a set $I \subseteq \Delta$;
- choose endpoints in set $\Delta_1 \backslash I$ subject to (2);
- test whether Δ_1 belongs to workingSet and whether Δ, Δ_1 do not overlap.

We claim that, in the above steps, the number of involved endpoints does not exceed $3f'$. To see this, we observe that from (2) we can derive $|I| \geq \varphi(\Delta) + \varphi(\Delta_1) - f'$. The total number of (distinct) endpoints in a single iteration step is $e = 2\varphi(\Delta) + 2\varphi(\Delta_1) - |I|$. Combining with the above inequality we have

$$\begin{aligned} e &\leq 2\varphi(\Delta) + 2\varphi(\Delta_1) - \varphi(\Delta) - \varphi(\Delta_1) + f' \\ &= \varphi(\Delta) + \varphi(\Delta_1) + f' \ \leq \ 3f' \ , \end{aligned}$$

as claimed. Since each endpoint takes values in the set $[0, n]$, we have a total of $\mathcal{O}(n^{3f'})$ different choices. For each such choice, we need to classify an endpoint as belonging to either $\Delta \backslash I$, $\Delta_1 \backslash I$, or I. This amounts to an additional $\mathcal{O}(3^{3f'})$ different choices. Overall, we have a total number of $\mathcal{O}((3n)^{3f'})$ different choices. For each such choice, the test for membership in workingSet for Δ_1 takes constant time in case we use a multi-dimensional table, or else $\mathcal{O}(|p|)$ in case we use a trie. The adjacency test and the merge operations can easily be carried out in time $\mathcal{O}(|p|)$.

Putting all of the above observations together, and using the already observed fact that $n = \mathcal{O}(|p|)$, we can conclude that the total amount of time required by the while-loop at lines 7 to 13 is bounded by $\mathcal{O}(|p| \cdot (3|p|)^{3f'})$, both under the assumption that workingSet is represented as a multi-dimensional table or as a trie. This is also a bound on the running time of the whole algorithm.

4.5 Minimal binarization of a complete LCFRS

The algorithm defined in Section 4.3 can be used to binarize an LCFRS in such a way that each rule in the resulting binarization has the minimum possible fan-out. This can be done by applying the BOUNDED-BINARIZATION algorithm to each production p, until we find the minimum value for the

```
1: Function MINIMAL-BINARIZATION($G$)
2: $p_b = \emptyset$ {Set of binarized productions}
3: for all production $p$ of $G$ do
4:     $f'$ = fan-out($p$);
5:     while not BOUNDED-BINARIZATION($p$, $f'$) do
6:         $f'$ = $f'$ + 1;
7:     add result of BOUNDED-BINARIZATION($p$, $f'$) to $p_b$; {We obtain the tree from BOUNDED-BINARIZATION as explained in Section 4.3 and use it to binarize $p$}
8: return $p_b$;
```

Figure 3: Minimal binarization by sequential search

bound f' for which this algorithm finds a binarization. For a production with rank r and fan-out f, we know that this optimal value of f' must be in the interval $[f, \lceil \frac{r}{2} \rceil \cdot f]$ because binarizing a production cannot reduce its fan-out, and the NAIVE-BINARIZATION algorithm seen in Section 4.1 can binarize any production by increasing fan-out to $\lceil \frac{r}{2} \rceil \cdot f$ in the worst case.

The simplest way of finding out the optimal value of f' for each production is by a sequential search starting with $\varphi(p)$ and going upwards, as in the algorithm in Figure 3. Note that the upper bound $\lceil \frac{r}{2} \rceil \cdot f$ that we have given for f' guarantees that the while-loop in this algorithm always terminates.

In the worst case, we may need $f \cdot (\lceil \frac{r}{2} \rceil - 1) + 1$ executions of the BOUNDED-BINARIZATION algorithm to find the optimal binarization of a production in G. This complexity can be reduced by changing the strategy to search for the optimal f': for example, we can perform a binary search within the interval $[f, \lceil \frac{r}{2} \rceil \cdot f]$, which lets us find the optimal binarization in $\lfloor \log(f \cdot (\lceil \frac{r}{2} \rceil - 1) + 1) \rfloor + 1$ executions of BOUNDED-BINARIZATION. However, this will not result in a practical improvement, since BOUNDED-BINARIZATION is exponential in the value of f' and the binary search will require us to run it on values of f' larger than the optimal in most cases. An intermediate strategy between the two is to apply exponential backoff to try the sequence of values $f-1+2^i$ (for $i = 0, 1, 2 \ldots$). When we find the first i such that BOUNDED-BINARIZATION does not fail, if $i > 0$, we apply the same strategy to the interval $[f-1+2^{i-1}, f-2+2^i]$, and we repeat this method to shrink the interval until BOUNDED-BINARIZATION does not fail for $i = 0$, giving us our optimal f'. With this strategy, the amount of executions of the algorithm that we need in the worst case is

$$\frac{1}{2}(\lceil \log(\omega) \rceil + \lceil \log(\omega) \rceil^2) + 1\,,$$

where $\omega = f \cdot (\lceil \frac{r}{2} \rceil - 1) + 1$, but we avoid using unnecessarily large values of f'.

5 Discussion

To conclude this paper, we now discuss a number of aspects of the results that we have presented, including various other pieces of research that are particularly relevant to this paper.

5.1 The tradeoff between rank and fan-out

The algorithm introduced in this paper can be used to transform an LCFRS into an equivalent form with rank 2. This will result into a more efficiently parsable LCFRS, since rank exponentially affects parsing complexity. However, we must take into account that parsing complexity is also influenced by fan-out. Our algorithm guarantees a minimal increase in fan-out. In practical cases it seems such an increase is quite small. For example, in the context of dependency parsing, both Gómez-Rodríguez et al. (2009) and Kuhlmann and Satta (2009) show that all the structures in several well-known non-projective dependency treebanks are binarizable without any increase in their fan-out.

More in general, it has been shown by Seki et al. (1991) that parsing of LCFRS can be carried out in time $\mathcal{O}(n^{|p_M|})$, where n is the length of the input string and p_M is the production in the grammar with largest size.[3] Thus, there may be cases in which one has to find an optimal tradeoff between rank and fan-out, in order to minimize the size of p_M. This requires some kind of Viterbi search over the space of all possible binarizations, constructed as described at the end of Subsection 4.3, for some appropriate value of the fan-out f'.

[3]The result has been shown for the formalism of multiple context-free grammars (MCFG), but it also applies to LCFRS, which are a special case of MCFG.

5.2 Extension to general LCFRS

This paper has focussed on *string-based* LCFRS. As discussed in Vijay-Shanker et al. (1987), LCFRS provide a more general framework where the productions are viewed as generating a set of abstract *derivation* trees. These trees can be used to specify how structures other than tuples of strings are composed. For example, LCFRS derivation trees can be used to specify how the elementary trees of a Tree Adjoining Grammar can be composed to produced derived tree. However, the results in this paper also apply to non-string-based LCFRS, since by limiting attention to the terminal string yield of whatever structures are under consideration, the composition operations can be defined using the string-based version of LCFRS that is discussed here.

5.3 Similar algorithmic techniques

The NAIVE-BINARIZATION algorithm given in Figure 1 is not novel to this paper: it is similar to an algorithm developed in Melamed et al. (2004) for generalized multitext grammars, a formalism weakly equivalent to LCFRS that has been introduced for syntax-based machine translation. However, the grammar produced by our algorithm has optimal (minimal) fan-out. This is an important improvement over the result in (Melamed et al., 2004), as this quantity enters into the parsing complexity of both multitext grammars and LCFRS as an exponential factor, and therefore must be kept as low as possible to ensure practically viable parsing.

Rank reduction is also investigated in Nesson et al. (2008) for synchronous tree-adjoining grammars, a synchronous rewriting formalism based on tree-adjoining grammars Joshi and Schabes (1992). In this case the search space of possible reductions is strongly restricted by the tree structures specified by the formalism, resulting in simplified computation for the reduction algorithms. This feature is not present in the case of LCFRS.

There is a close parallel between the technique used in the MINIMAL-BINARIZATION algorithm and deductive parsing techniques as proposed by Shieber et al. (1995), that are usually implemented by means of tabular methods. The idea of exploiting tabular parsing in production factorization was first expressed in Zhang et al. (2006). In fact, the particular approach presented here has been used to improve efficiency of parsing algorithms that use discontinuous syntactic models, in particular, non-projective dependency grammars, as discussed in Gómez-Rodríguez et al. (2009).

5.4 Open problems

The bounded binarization algorithm that we have presented has exponential run-time in the value of the input fan-out bound f'. It remains an open question whether the bounded binarization problem for LCFRS can be solved in deterministic polynomial time. Even in the restricted case of $f' = \varphi(p)$, that is, when no increase in the fan-out of the input production is allowed, we do not know whether p can be binarized using only deterministic polynomial time in the value of p's fan-out. However, our bounded binarization algorithm shows that the latter problem can be solved in polynomial time when the fan-out of the input LCFRS is bounded by some constant.

Whether the bounded binarization problem can be solved in polynomial time in the value of the input bound f' is also an open problem in the restricted case of synchronous context-free grammars, a special case of an LCFRS of fan-out two with a strict separation between the two components of each nonterminal in the right-hand side of a production, as discussed in the introduction. An interesting analysis of this restricted problem can be found in Gildea and Stefankovic (2007).

Acknowledgements The work of Carlos Gómez-Rodríguez was funded by Ministerio de Educación y Ciencia and FEDER (HUM2007-66607-C04) and Xunta de Galicia (PGIDIT07SIN005206PR, INCITE08E1R104022ES, INCITE08ENA305025ES, INCITE08PXIB302179PR and Rede Galega de Procesamento da Linguaxe e Recuperación de Información). The work of Marco Kuhlmann was funded by the Swedish Research Council. The work of Giorgio Satta was supported by MIUR under project PRIN No. 2007TJNZRE_002. We are grateful to an anonymous reviewer for a very detailed review with a number of particularly useful suggestions.

References

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Daniel Gildea and Daniel Stefankovic. 2007. Worst-case synchronous grammar rules. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 147–154. Association for Computational Linguistics, Rochester, New York.

Carlos Gómez-Rodríguez, David J. Weir, and John Carroll. 2009. Parsing mildly non-projective dependency structures. In *Twelfth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. To appear.

A. K. Joshi and Y. Schabes. 1992. Tree adjoining grammars and lexicalized grammars. In M. Nivat and A. Podelsky, editors, *Tree Automata and Languages*. Elsevier, Amsterdam, The Netherlands.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL), Main Conference Poster Sessions*, pages 507–514. Sydney, Australia.

Marco Kuhlmann and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Twelfth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. To appear.

I. Dan Melamed, Benjamin Wellington, and Giorgio Satta. 2004. Generalized multitext grammars. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 661–668. Barcelona, Spain.

Rebecca Nesson, Giorgio Satta, and Stuart M. Shieber. 2008. Optimal k-arization of synchronous tree-adjoining grammar. In *Proceedings of ACL-08: HLT*, pages 604–612. Association for Computational Linguistics, Columbus, Ohio.

A. Nijholt. 1980. *Context-Free Grammars: Covers, Normal Forms, and Parsing*, volume 93. Springer-Verlag, Berlin, Germany.

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

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

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

Takeaki Uno and Mutsunori Yagiura. 2000. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309.

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

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 977–984. Sydney, Australia.

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.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 256–263. New York, USA.