Parsing Linear Context-Free Rewriting Systems

Håkan Burden
Dept. of Linguistics
Göteborg University
cl1hburd@cling.gu.se

Peter Ljunglöf
Dept. of Computing Science
Göteborg University
peb@cs.chalmers.se

Abstract

We describe four different parsing algorithms for Linear Context-Free Rewriting Systems (Vijay-Shanker et al., 1987). The algorithms are described as deduction systems, and possible optimizations are discussed.

The only parsing algorithms presented for *linear context-free rewriting systems* (LCFRS; Vijay-Shanker et al., 1987) and the equivalent formalism *multiple context-free grammar* (MCFG; Seki et al., 1991) are extensions of the CKY algorithm (Younger, 1967), more designed for their theoretical interest, and not for practical purposes. The reason for this could be that there are not many implementations of these grammar formalisms. However, since a very important subclass of the Grammatical Framework (Ranta, 2004) is equivalent to LCFRS/MCFG (Ljunglöf, 2004a; Ljunglöf, 2004b), there is a need for practical parsing algorithms.

In this paper we describe four different parsing algorithms for Linear Context-Free Rewriting Systems. The algorithms are described as deduction systems, and possible optimizations are discussed.

1 Introductory definitions

A *record* is a structure $\Gamma = \{r_1 = a_1; \ldots; r_n = a_n\}$, where all r_i are distinct. That this can be seen as a set of feature-value pairs. This means that we can define a simple version of *record unification* $\Gamma_1 \sqcup \Gamma_2$ as the union $\Gamma_1 \cup \Gamma_2$, provided that there is no r such that $\Gamma_1.r \neq \Gamma_2.r$.

We sometimes denote a sequence $X_1, \ldots, X_n$ by the more compact $\vec{X}$. To update the ith record in a list of records, we write $\vec{\Gamma}[i := \Gamma]$. To substitute a variable B_k for a record Γ_k in any data structure Γ, we write $\Gamma[B_k/\Gamma_k]$.

1.1 Decorated Context-Free Grammars

The context-free approximation described in section 4 uses a form of CFG with decorated rules of the form $f : A \to \alpha$, where f is the name of the rule, and α is a sequence of terminals and categories subscripted with information needed for post-processing of the context-free parse result. In all other respects a decorated CFG can be seen as a straight-forward CFG.

1.2 Linear Context-Free Rewriting Systems

A *linear context-free rewriting system* (LCFRS; Vijay-Shanker et al., 1987) is a linear, non-erasing *multiple context-free grammar* (MCFG; Seki et al., 1991). An MCFG rule is written[1]

$$A \to f[B_1 \ldots B_\delta] \quad := \quad \{\, r_1 = \alpha_1; \ldots; r_n = \alpha_n \,\}$$

where A and B_i are categories, f is the name of the rule, r_i are record labels and α_i are sequences of terminals and argument projections of the form $B_i.r$. The *language* $\mathcal{L}(A)$ of a category A is a set of string records, and is defined recursively as

$$\begin{aligned} \mathcal{L}(A) \quad = \quad \{ \quad & \Phi[B_1/\Gamma_1, \ldots, B_\delta/\Gamma_\delta] \quad | \\ & A \to f[B_1 \ldots B_\delta] := \Phi, \\ & \Gamma_1 \in \mathcal{L}(B_1),\ \ldots,\ \Gamma_\delta \in \mathcal{L}(B_\delta) \quad \} \end{aligned}$$

It is the possibility of discontinuous constituents that makes LCFRS/MCFG more expressive than context-free grammars. If the grammar only consists of single-label records, it generates a context-free language.

Example A small example grammar is shown in figure 1, and generates the language

$$\mathcal{L}(S) \quad = \quad \{\, s\, s_{hm} \mid s \in (a \cup b)^* \,\}$$

where s_{hm} is the homomorphic mapping such that each a in s is translated to c, and each b is translated to d. Examples of generated strings are ac, $abcd$ and $bbaddc$. However, neither abc nor $abcdabcd$ will be

[1] We borrow the idea of equating argument categories and variables from Nakanishi et al. (1997) , but instead of tuples we use the equivalent notion of records for the linearizations.

Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT), pages 11–17,
Vancouver, October 2005.

Figure 1: An example grammar describing the language $\{ s\, s_{hm} \mid s \in (a \cup b)^* \}$

$$
\begin{array}{rcl}
S \rightarrow f[A] & := & \{ s = A.p\ A.q \} \\
A \rightarrow g[A_1\ A_2] & := & \{ p = A_1.p\ A_2.p;\ q = A_1.q\ A_2.q \} \\
A \rightarrow ac[\,] & := & \{ p = a;\ q = c \} \\
A \rightarrow bd[\,] & := & \{ p = b;\ q = d \}
\end{array}
$$

generated. The language is not context-free since it contains a combination of multiple and crossed agreement with duplication.

If there is at most one occurrence of each possible projection $A_i.r$ in a linearization record, the MCFG rule is *linear*. If all rules are linear the grammar is linear. A rule is *erasing* if there are argument projections that have no realization in the linearization. A grammar is erasing if it contains an erasing rule. It is possible to transform an erasing grammar to non-erasing form (Seki et al., 1991).

Example The example grammar is both linear and non-erasing. However, given that grammar, the rule

$$E \rightarrow e[A] \quad := \quad \{ r_1 = A.p;\ r_2 = A.p \}$$

is both non-linear (since $A.p$ occurs more than once) and erasing (since it does not mention $A.q$).

1.3 Ranges

Given an input string w, a *range* ρ is a pair of indices, (i, j) where $0 \leq i \leq j \leq |w|$ (Boullier, 2000). The entire string $w = w_1 \ldots w_n$ spans the range $(0, n)$. The word w_i spans the range $(i - 1, i)$ and the substring $w_{i+1}, \ldots, w_j$ spans the range (i, j). A range with identical indices, (i, i), is called an empty range and spans the empty string.

A record containing label-range pairs,

$$\Gamma \quad = \quad \{ r_1 = \rho_1, \ldots, r_n = \rho_n \}$$

is called a *range record*. Given a range $\rho = (i, j)$, the *ceiling* of ρ returns an empty range for the right index, $\lceil \rho \rceil = (j, j)$; and the *floor* of ρ does the same for the left index $\lfloor \rho \rfloor = (i, i)$. Concatenation of two ranges is non-deterministic,

$$(i, j) \cdot (j', k) \quad = \quad \{ (i, k) \mid j = j' \}$$

.

1.3.1 Range restriction

In order to retrieve the ranges of any substring s in a sentence $w = w_1 \ldots w_n$ we define *range restriction* of s with respect to w as $\langle s \rangle^w = \{ (i, j) \mid s = w_{i+1} \ldots w_j \}$, i.e. the set of all occurrences of s in w. If w is understood from the context we simply write $\langle s \rangle$.

Range restriction of a linearization record Φ is written $\langle \Phi \rangle$, which is a set of records, where every terminal token s is replaced by a range from $\langle s \rangle$. The range restriction of two terminals next to each other fails if range concatenation fails for the resulting ranges. Any unbound variables in Φ are unaffected by range restriction.

Example Given the string $w = abba$, range restricting the terminal a yields

$$\langle a \rangle^w \quad = \quad \{ (0, 1),\ (3, 4) \}$$

Furthermore,

$$\langle a\ A.r\ a\ b\ B.q \rangle^w \quad = \quad \{ (0, 1)\ A.r\ (0, 2)\ B.q,\ (3, 4)\ A.r\ (0, 2)\ B.q \}$$

The other possible solutions fail since they cannot be range concatenated.

2 Parsing as deduction

The idea with *parsing as deduction* (Shieber et al., 1995) is to deduce parse items by inference rules. A parse item is a representation of a piece of information that the parsing algorithm has acquired. An inference rule is written

$$\frac{\gamma_1 \ldots \gamma_n \quad C}{\gamma}$$

where γ is the consequence of the antecedents $\gamma_1 \ldots \gamma_n$, given that the side conditions in C hold.

2.1 Parsing decorated CFG

Decorated CFG can be parsed in a similar way as standard CFG. For our purposes it suffices to say that the algorithm returns items of the form,

$$[f : A/\rho \rightarrow B_1/\rho_1\ \ldots\ B_n/\rho_n \bullet\,]$$

saying that A spans the range ρ, and each daughter B_i spans ρ_i.

The standard inference rule *combine* might look like this for decorated CFG:

Combine

$$\frac{\begin{array}{l}[f : A/\rho \rightarrow \alpha \bullet B_x\ \beta] \\ [g : B/\rho' \rightarrow \ldots \bullet\,] \\ \rho'' \in \rho \cdot \rho'\end{array}}{[f : A/\rho \rightarrow \alpha\ B_x/\rho'' \bullet\, \beta]}$$

Note that the subscript x in B_x is the decoration that will only be used in post-processing.

3 The Naïve algorithm

Seki et al. (1991) give an algorithm for MCFG, which can be seen as an extension of the CKY algorithm (Younger, 1967). The problem with that algorithm is that it has to find items for all daughters at the same time. We modify this basic algorithm to be able to find one daughter at the time.

There are two kinds of items. A *passive* item $[A;\ \Gamma]$ has the meaning that the category A has been found spanning the range record Γ. An *active* item for the rule $A \to f[\vec{B}\ \vec{B'}] := \Psi$ has the form

$$[A \to f[\vec{B} \bullet \vec{B'}];\ \Phi;\ \vec{\Gamma}]$$

in which the categories to the left of the dot, $\vec{B}$, have been found with the linearizations in the list of range records $\vec{\Gamma}$. Φ is the result of substituting the projections in Ψ with ranges for the categories found in $\vec{B}$.

3.1 Inference rules

There are three inference rules, Predict, Combine and Convert.

Predict

$$\frac{\begin{array}{l} A \to f[\vec{B}] := \Psi \\ \Phi \in \langle \Psi \rangle \end{array}}{[A \to f[\ \bullet \vec{B}];\ \Phi;\ ]}$$

Prediction gives an item for every rule in the grammar, where the range restriction Φ is what has been found from the beginning. The list of daughters is empty since none of the daughters in $\vec{B}$ have been found yet.

Combine

$$\frac{\begin{array}{l} [A \to f[\vec{B} \bullet B_k\, \vec{B'}];\ \Phi;\ \vec{\Gamma}] \\ [B_k;\ \Gamma_k] \\ \quad \Phi' \in \Phi[B_k/\Gamma_k] \end{array}}{[A \to f[\vec{B}\, B_k \bullet \vec{B'}];\ \Phi';\ \vec{\Gamma},\ \Gamma_k]}$$

An active item looking for B_k and a passive item that has found B_k can be combined into a new active item. In the new item we substitute B_k for Γ_k in the linearization record. We also add Γ_k to the new item's list of daughters.

Convert

$$\frac{\begin{array}{l} [A \to f[\vec{B} \bullet\ ];\ \Phi;\ \vec{\Gamma}] \\ \quad \Gamma \equiv \Phi \end{array}}{[A;\ \Gamma]}$$

Every fully instantiated active item is converted into a passive item. Since the linearization record Φ is fully instantiated, it is equivalent to the range record Γ.

Figure 2: The example grammar converted to a decorated CFG

$$\begin{array}{rcl} f : (S.s) & \to & (A.p)\ (A.q) \\ g : (A.p) & \to & (A.p)_1\ (A.p)_2 \\ g : (A.q) & \to & (A.q)_1\ (A.q)_2 \\ ac : (A.p) & \to & a \\ ac : (A.q) & \to & b \\ bd : (A.p) & \to & c \\ bd : (A.q) & \to & d \end{array}$$

The subscripted numbers are for distinguishing the two categories from each other, since they are equivalent. Here $A.q$ is a context-free category of its own, not a record projection.

4 The Approximative algorithm

Parsing is performed in two steps in the approximative algorithm. First we parse the sentence using a context-free approximation. Then the resulting context-free chart is recovered to a LCFRS chart.

The LCFRS is converted by creating a decorated context-free rule for every row in a linearization record. Thus, the rule

$$A \to f[\vec{B}] \quad := \quad \{\, r_1 = \alpha_1;\ \ldots;\ r_n = \alpha_n \,\}$$

will give n context-free rules $f : A.r_i \to \alpha_i$. The example grammar from figure 1 is converted to a decorated CFG in figure 2.

Parsing is now initiated by a context-free parsing algorithm returning decorated items as in section 2.1. Since the categories of the decorated grammar are projections of LCFRS categories, the final items will be of the form

$$[f : (A.r)/\rho \to \ldots (B.r')_x/\rho' \ldots \bullet\ ]$$

Since the decorated CFG is over-generating, the returned parse chart is unsound. We therefore need to retrieve the items from the decorated CFG parse chart and check them against the LCFRS to get the discontinuous constituents and mark them for validity.

The *initial parse items* are of the form,

$$[A \to f[\vec{B}];\ r = \rho;\ \vec{\Gamma}]$$

where $\vec{\Gamma}$ is extracted from a corresponding decorated item $[f : (A.r)/\rho \to \beta]$, by partitioning the daughters in β such that $\Gamma_i = \{\, r = \rho \mid (B.r)_i/\rho \in \beta \,\}$. In other words, Γ_i will consist of all $r = \rho$ such that $B.r$ is subscripted by i in the decorated item.

Example Given $\beta = (A.p)_2/\rho'\ (B.q)_1/\rho''\ (A.q)_2/\rho'''$, we get the two range records $\Gamma_1 = \{q = \rho''\}$ and $\Gamma_2 = \{p = \rho';\ q = \rho'''\}$.

Apart from the initial items, we use three kinds of parse items. From the initial parse items we first build *LCFRS items*, of the form

$$[A \to f[\vec{B}];\ \Gamma \bullet r_i \dots r_n;\ \vec{\Gamma}]$$

where $r_i \dots r_n$ is a list of labels, $\vec{\Gamma}$ is a list of $|\vec{B}|$ range records, and Γ is a range record for the labels $r_1 \dots r_{i-1}$.

In order to recover the chart we use *mark items*

$$[A \to f[\vec{B} \bullet \vec{B}'];\ \Gamma;\ \vec{\Gamma} \bullet \vec{\Gamma}']$$

The idea is that $\vec{\Gamma}$ has been verified as range records spanning the daughters $\vec{B}$. When all daughters have been verified, a mark item is converted to a *passive item* $[A;\ \Gamma]$.

4.1 Inference rules

There are five inference rules, Pre-Predict, Pre-Combine, Mark-Predict, Mark-Combine and Convert.

Pre-Predict

$$\frac{\begin{array}{l} A \to f[\vec{B}] := \{r_1 = \alpha_1; \dots; r_n = \alpha_n\} \\ \vec{\Gamma}_\delta = \{\,\}, \dots, \{\,\} \end{array}}{[A \to f[\vec{B}];\ \bullet\, r_1 \dots r_n;\ \vec{\Gamma}_\delta]}$$

Every rule $A \to f[\vec{B}]$ is predicted as an LCFRS item. Since the context-free items contain information about $\alpha_1 \dots \alpha_n$, we only need to use the labels $r_1, \dots, r_n$. $\vec{\Gamma}_\delta$ is a list of $|\vec{B}|$ empty range records.

Pre-Combine

$$\frac{\begin{array}{l} [R;\ \Gamma \bullet r\, r_i \dots r_n;\ \vec{\Gamma}] \\ [R;\ r = \rho;\ \vec{\Gamma}'] \\ \vec{\Gamma}'' \in \vec{\Gamma} \sqcup \vec{\Gamma}' \end{array}}{[R;\ \{\Gamma; r = \rho\} \bullet r_i \dots r_n;\ \vec{\Gamma}'']}$$

If there is an initial parse item for the rule R with label r, we can combine it with an LCFRS item looking for r, provided the daughters' range records can be unified.

Mark-Predict

$$\frac{[A \to [\vec{B}];\ \Gamma \bullet\,;\ \vec{\Gamma}]}{[A \to [\,\bullet \vec{B}];\ \Gamma;\ \bullet\, \vec{\Gamma}]}$$

When all record labels have been found, we can start to check if the items have been derived in a valid way by marking the daughters' range records for correctness.

Mark-Combine

$$\frac{\begin{array}{l} [A \to f[\vec{B} \bullet B_i\, \vec{B}'];\ \Gamma;\ \vec{\Gamma} \bullet \Gamma_i\, \vec{\Gamma}'] \\ [B_i;\ \Gamma_i] \end{array}}{[A \to f[\vec{B}\, B_i \bullet \vec{B}'];\ \Gamma;\ \vec{\Gamma}\, \Gamma_i \bullet \vec{\Gamma}']}$$

Record Γ_i is correct if there is a correct passive item for category B_i that has found Γ_i.

Convert

$$\frac{[A \to f[\vec{B} \bullet];\ \Gamma;\ \vec{\Gamma} \bullet]}{[A;\ \Gamma]}$$

An item that has marked all daughters as correct is converted to a passive item.

5 The Active algorithm

The active algorithm parses without using any context-free approximation. Compared to the Naïve algorithm the dot is used to traverse the linearization record of a rule instead of the categories in the right-hand side.

For this algorithm we use a special kind of range, ρ^ϵ, which denotes simultaneously all empty ranges (i, i). Range restricting the empty string gives $\langle\epsilon\rangle = \rho^\epsilon$. Concatenation is defined as $\rho{\cdot}\rho^\epsilon = \rho^\epsilon{\cdot}\rho = \rho$. Both the ceiling and the floor of ρ^ϵ are identities, $\lceil\rho^\epsilon\rceil = \lfloor\rho^\epsilon\rfloor = \rho^\epsilon$.

There are two kinds of items. *Passive items* $[A;\ \Gamma]$ say that we have found category A inside the range record Γ. An *active item* for the rule

$$A \to f[\vec{B}] \quad := \quad \{\Phi;\ r = \alpha\beta;\ \Psi\}$$

is of the form

$$[A \to f[\vec{B}];\ \Gamma,\ r = \rho \bullet \beta,\ \Psi;\ \vec{\Gamma}]$$

where Γ is a range record corresponding to the linearization rows in Φ and α has been recognized spanning ρ. We are still looking for the rest of the row, β, and the remaining linearization rows Ψ. $\vec{\Gamma}$ is a list of range records containing information about the daughters $\vec{B}$.

5.1 Inference rules

There are five inference rules, Predict, Complete, Scan, Combine and Convert.

Predict

$$\frac{\begin{array}{l} A \to f[\vec{B}] := \{r = \alpha; \Phi\} \\ \vec{\Gamma}_\delta = \{\,\}, \dots, \{\,\} \end{array}}{[A \to f[\vec{B}];\ \{\},\ r = \rho^\epsilon \bullet \alpha,\ \Phi;\ \vec{\Gamma}_\delta]}$$

For every rule in the grammar, predict a corresponding item that has found the empty range. $\vec{\Gamma}_\delta$ is a list of $|\vec{B}|$ empty range records since nothing has been found yet.

Complete

$$\frac{[R;\ \Gamma,\ r = \rho \bullet \epsilon,\ \{r' = \alpha; \Phi\};\ \vec{\Gamma}]}{[R;\ \{\Gamma; r = \rho\},\ r' = \rho^\epsilon \bullet \alpha, \Phi;\ \vec{\Gamma}]}$$

When an item has found an entire linearization row we continue with the next row by starting it off with the empty range.

Scan

$$\frac{[R;\, \Gamma,\, r = \rho \bullet s\,\alpha,\, \Phi;\, \vec{\Gamma}] \quad \rho' \in \rho \cdot \langle s \rangle}{[R;\, \Gamma,\, r = \rho' \bullet \alpha,\, \Phi;\, \vec{\Gamma}]}$$

When the next symbol to read is a terminal, its range restriction is concatenated with the range for what the row has found so far.

Combine

$$\frac{[A \to f[\vec{B}];\, \Gamma,\, r = \rho \bullet B_i.r'\,\alpha,\, \Phi;\, \vec{\Gamma}] \quad [B_i;\, \Gamma'] \quad \rho' \in \rho \cdot \Gamma'.r' \quad \Gamma_i \subseteq \Gamma'}{[A \to f[\vec{B}];\, \Gamma,\, r = \rho' \bullet \alpha,\, \Phi;\, \vec{\Gamma}[i := \Gamma']]}$$

If the next thing to find is a projection on B_i, and there is a passive item where B_i is the category, where Γ' is consistent with Γ_i, we can move the dot past the projection. Γ_i is updated with Γ', since it might contain more information about the ith daughter.

Convert

$$\frac{[A \to f[\vec{B}];\, \Gamma,\, r = \rho \bullet \epsilon,\, \{\};\, \vec{\Gamma}]}{[A;\, \{\Gamma; r = \rho\}]}$$

An active item that has fully recognized all its linearization rows is converted to a passive item.

6 The Incremental algorithm

An incremental algorithm reads one token at the time and calculates all possible consequences of the token before the next token is read[2]. The Active algorithm as described above is not incremental, since we do not know in which order the linearization rows of a rule are recognized. To be able to parse incrementally, we have to treat the linearization records as sets of feature-value pairs, instead of a sequence.

The items for a rule $A \to f[\vec{B}] := \Phi$ have the same form as in the Active algorithm:

$$[A \to f[\vec{B}];\, \Gamma,\, r = \rho \bullet \beta,\, \Psi;\, \vec{\Gamma}]$$

However, the order between the linearization rows does not have to be the same as in Φ. Note that in this algorithm we do not use passive items. Also note that since we always know where in the input we are, we cannot make use of a distinguished ϵ-range. Another consequence of knowing the current input position is that there are fewer possible matches for the Combine rule.

[2]See e.g. the ACL 2004 workshop "Incremental Parsing: Bringing Engineering and Cognition Together".

6.1 Inference rules

There are four inference rules, Predict, Complete, Scan and Combine.

Predict

$$\frac{A \to f[\vec{B}] := \{\Phi; r = \alpha; \Psi\} \quad 0 \leq k \leq |w|}{[A \to f[\vec{B}];\, \{\},\, r = (k,k) \bullet \alpha,\, \{\Phi; \Psi\};\, \vec{\Gamma}_\delta]}$$

An item is predicted for every linearization row r and every input position k. $\vec{\Gamma}_\delta$ is a list of $|\vec{B}|$ empty range records.

Complete

$$\frac{[R;\, \Gamma,\, r = \rho \bullet \epsilon,\, \{\Phi; r' = \alpha; \Psi\};\, \vec{\Gamma}] \quad \lceil \rho \rceil \leq k \leq |w|}{[R;\, \{\Gamma; r = \rho\},\, r' = (k,k) \bullet \alpha,\, \{\Phi; \Psi\};\, \vec{\Gamma}]}$$

Whenever a linearization row r is fully traversed, we predict an item for every remaining linearization row r' and every remaining input position k.

Scan

$$\frac{[R;\, \Gamma,\, r = \rho \bullet s\,\alpha,\, \Phi;\, \vec{\Gamma}] \quad \rho' \in \rho \cdot \langle s \rangle}{[R;\, \Gamma,\, r = \rho' \bullet \alpha,\, \Phi;\, \vec{\Gamma}]}$$

If the next symbol in the linearization row is a terminal, its range restriction is concatenated to the range for the partially recognized row.

Combine

$$\frac{[R;\, \Gamma,\, r = \rho \bullet B_i.r'\,\alpha,\, \Phi;\, \vec{\Gamma}] \quad [B_i \to \ldots;\, \Gamma',\, r' = \rho' \bullet \epsilon,\, \ldots;\, \ldots] \quad \rho'' \in \rho \cdot \rho' \quad \Gamma_i \subseteq \{\Gamma'; r' = \rho'\}}{[R;\, \Gamma,\, r = \rho'' \bullet \alpha,\, \Phi;\, \vec{\Gamma}[i := \{\Gamma'; r' = \rho'\}]]}$$

If the next item is a record projection $B_i.r'$, and there is an item for B_i which has found r', then move the dot forward. The information in Γ_i must be consistent with the information found for the B_i item, $\{\Gamma'; r' = \rho'\}$.

7 Discussion

We have presented four different parsing algorithms for LCFRS/MCFG. The algorithms are described as deduction systems, and in this final section we discuss some possible optimizations, and complexity issues.

7.1 Different prediction strategies

The Predict rule in the above described algorithms is very crude, predicting an item for each rule in the grammar (for the Incremental algorithm even for each input position). A similar context-free prediction rule is called *bottom-up Earley* by Sikkel and Nijholt (1997). Such crude predictions are only intended for educational purposes, since they lead to lots of uninteresting items, and waste of computing power. For practical purposes there are two standard context-free prediction strategies, top-down and bottom-up (see e.g. Wirén (1992)) and they can be adapted to the algorithms presented in this paper.

The main idea is that an item for the rule $A \to f[\vec{B}]$ with the linearization row $r = \alpha$ is only predicted if...

(Top-down prediction) ...there is another item looking for $A.r$.

(Bottom-up prediction) ...there is an passive item that has found the first symbol in α.

For a more detailed description of these prediction strategies, see Ljunglöf (2004a).

7.2 Efficiency and complexity of the algorithms

The theoretical time complexity for these algorithms is not better than what has been presented earlier.[3] The complexity arguments are similar and the reader is referred to Seki et al. (1991).

However, theoretical time complexity does not say much about practical performance, as is already clear from context-free parsing, where the theoretical time complexity has remained the same ever since the first publications (Kasami, 1965; Younger, 1967). There are two main ways of improving the efficiency of existing algorithms, which can be called *refinement* and *filtering* (Sikkel and Nijholt, 1997). First, one wants to be able to locate existing parse items efficiently, e.g. by indexing some properties in a hash table. This is often done by *refining* the parse items or inference rules, increasing the number of items or deduction steps. Second, it is desirable to reduce the number of parse items, which can be done by *filtering* out redundant parts of an algorithm.

The algorithms presented in this paper can all be seen as refinements and filterings of the basic algorithm of Seki et al. (1991):

The naïve algorithm is a refinement of the basic algorithm, since single items and deduction steps are decomposed into several different items and smaller deduction steps.

The approximative algorithm is both a refinement and a filtering of the naïve algorithm; a refinement since the inference rules Pre-Predict and Pre-Combine are added, and a filtering since there will hopefully be less items for Mark-Predict and Mark-Combine to take care of.

The active algorithm is a refinement of the naïve algorithm, since the Combine rule is divided into the rules Complete, Scan and Combine.

The incremental algorithm is finally a refinement of the active algorithm, since Predict and Complete can select from any possible remaining linearization row, and not just the following.

Furthermore, the different prediction strategies (top-down and bottom-up), become filterings of the algorithms, since they reduce the number of parse items.

7.3 Implementing and testing the algorithms

The algorithms presented in this paper have been implemented in the programming language Haskell, for inclusion in the Grammatical Framework system (Ranta, 2004). These implementations are described by Burden (2005). We have also started to implement a selection of the algorithms in the programming language Prolog.

Preliminary results suggest that the Active algorithm with bottom-up prediction is a good candidate for parsing grammars written in the Grammatical Framework. For a normal sentence in the English resource grammar the speedup is about 20 times when compared to context-free parsing and filtering of the parse trees. In the future we plan to test the different algorithms more extensively.

Acknowledgments

The authors are supported by the EU project TALK (Talk and Look, Tools for Ambient Linguistic Knowledge), IST-507802.

References

Pierre Boullier. 2000. Range concatenation grammars. In *6th International Workshop on Parsing Technologies*, pages 53–64, Trento, Italy.

Håkan Burden. 2005. Implementations of parsing algorithms for linear multiple context-free grammars. Master's thesis, Göteborg University, Gothenburg, Sweden.

Tadao Kasami. 1965. An efficient recognition and syntax algorithm for context-free languages. Technical Report AFCLR-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

[3] Nakanishi et al. (1997) reduce the parsing problem to boolean matrix multiplication, but this can be considered a purely theoretical result.

Peter Ljunglöf. 2004a. *Expressivity and Complexity of the Grammatical Framework*. Ph.D. thesis, Göteborg University and Chalmers University of Technology, Gothenburg, Sweden.

Peter Ljunglöf. 2004b. Grammatical Framework and multiple context-free grammars. In *9th Conference on Formal Grammar*, Nancy, France.

Ryuichi Nakanishi, Keita Takada, and Hiroyuki Seki. 1997. An efficient recognition algorithm for multiple context-free languages. In *MOL5: 5th Meeting on the Mathematics of Language*, pages 119–123, Saarbrücken, Germany.

Aarne Ranta. 2004. Grammatical Framework, a type-theoretical grammar formalism. *Journal of Functional Programming*, 14(2):145–189.

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

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

Klaas Sikkel and Anton Nijholt. 1997. Parsing of context-free languages. In G. Rozenberg and A. Salomaa, editors, *The Handbook of Formal Languages*, volume II, pages 61–100. Springer-Verlag, Berlin.

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

Mats Wirén. 1992. *Studies in Incremental Natural-Language Analysis*. Ph.D. thesis, Linköping University, Linköping, Sweden.

Daniel H Younger. 1967. Recognition of context-free languages in time n^3. *Information and Control*, 10(2):189–208.