An Earley Parsing Algorithm for Range Concatenation Grammars

Laura Kallmeyer  
SFB 441  
Universität Tübingen  
72074 Tübingen, Germany  
lk@sfs.uni-tuebingen.de

Wolfgang Maier  
SFB 441  
Universität Tübingen  
72074 Tübingen, Germany  
wo.maier@uni-tuebingen.de

Yannick Parmentier  
CNRS - LORIA  
Nancy Université  
54506 Vandoeuvre, France  
parmenti@loria.fr

Abstract

We present a CYK and an Earley-style algorithm for parsing Range Concatenation Grammar (RCG), using the deductive parsing framework. The characteristic property of the Earley parser is that we use a technique of range boundary constraint propagation to compute the yields of non-terminals as late as possible. Experiments show that, compared to previous approaches, the constraint propagation helps to considerably decrease the number of items in the chart.

1 Introduction

RCGs (Boullier, 2000) have recently received a growing interest in natural language processing (Søgaard, 2008; Sagot, 2005; Kallmeyer et al., 2008; Maier and Søgaard, 2008). RCGs generate exactly the class of languages parsable in deterministic polynomial time (Bertsch and Nederhof, 2001). They are in particular more powerful than linear context-free rewriting systems (LCFRS) (Vijay-Shanker et al., 1987). LCFRS is unable to describe certain natural language phenomena that RCGs actually can deal with. One example is long-distance scrambling phenomena (Becker et al., 1991; Becker et al., 1992). Other examples are non-semilinear constructions such as case stacking in Old Georgian (Michaelis and Kracht, 1996) and Chinese number names (Radzinski, 1991). Boullier (1999) shows that RCGs can describe the permutations occurring with scrambling and the construction of Chinese number names.

Parsing algorithms for RCG have been introduced by Boullier (2000), who presents a directional top-down parsing algorithm using pseudocode, and Barthélémy et al. (2001), who add an oracle to Boullier’s algorithm. The more restricted class of LCFRS has received more attention concerning parsing (Villemonde de la Clergerie, 2002; Burden and Ljunglöf, 2005). This article proposes new CYK and Earley parsers for RCG, formulating them in the framework of parsing as deduction (Shieber et al., 1995). The second section introduces necessary definitions. Section 3 presents a CYK-style algorithm and Section 4 extends this with an Earley-style prediction.

2 Preliminaries

The rules (clauses) of RCGs1 rewrite predicates ranging over parts of the input by other predicates. E.g., a clause $S(aXb) \rightarrow S(X)$ signifies that $S$ is true for a part of the input if this part starts with an $a$, ends with a $b$, and if, furthermore, $S$ is also true for the part between $a$ and $b$.

Definition 1. A RCG $G = (N,T,V,P,S)$ consists of a) a finite set of predicates $N$ with an arity function $\text{dim}: N \rightarrow \mathbb{N} \setminus \{0\}$ where $S \in N$ is the start predicate with $\text{dim}(S) = 1$, b) disjoint finite sets of terminals $T$ and variables $V$, c) a finite set $P$ of clauses $\psi_0 \rightarrow \psi_1 \ldots \psi_m$ where $m \geq 0$ and each of the $\psi_i, 0 \leq i \leq m$, is a predicate of the form $A_i(\alpha_1, \ldots, \alpha_{\text{dim}(A_i)})$ with $A_i \in N$ and $\alpha_j \in (T \cup V)^* \text{ for } 1 \leq j \leq \text{dim}(A_i)$.

Central to RCGs is the notion of ranges on strings.

Definition 2. For every $w = w_1 \ldots w_n$ with $w_i \in T$ ($1 \leq i \leq n$), we define a) $\text{Pos}(w) = \{0, \ldots, n\}$. b) $(l,r) \in \text{Pos}(w) \times \text{Pos}(w)$ with $l \leq r$ is a range in $w$. Its yield $(l,r)\langle w \rangle$ is the substring $w_l \ldots w_r$. c) For two ranges $p_1 = \langle l_1, r_1 \rangle, p_2 = \langle l_2, r_2 \rangle$: if $r_1 = l_2$, then $p_1 \cdot p_2 = \langle l_1, r_2 \rangle$; otherwise $p_1 \cdot p_2$ is undefined. d) A vector $\phi = (\langle x_1, y_1 \rangle, \ldots, \langle x_k, y_k \rangle)$ is a range vector of dimension $k$ in $w$ if $\langle x_i, y_i \rangle$ is a range in $w$ for $1 \leq i \leq k$. $\phi(i).l$ (resp. $\phi(i).r$) denotes then the

---

1 In this paper, by RCG, we always mean positive RCG, see Boullier (2000) for details.
first (resp. second) component of the \(i\)th element of \(\phi\), that is \(x_i\) (resp. \(y_i\)).

In order to instantiate a clause of the grammar, we need to find ranges for all variables in the clause and for all occurrences of terminals. For convenience, we assume the variables in a clause and the occurrences of terminals to be equipped with distinct subscript indices, starting with 1 and ordered from left to right (where for variables, only the first occurrence is relevant for this order).

We introduce a function \(\Upsilon : P \rightarrow \mathbb{N}\) that gives the maximal index in a clause, and we define \(\Upsilon(c,x)\) for a given clause \(c\) and \(x\) a variable or an occurrence of a terminal as the index of \(x\) in \(c\).

**Definition 3.** An instantiation of a \(c \in P\) with \(\Upsilon(c) = j\) w.r.t. some string \(w\) is given by a range vector \(\phi\) of dimension \(j\). Applying \(\phi\) to a predicate \(A(\vec{\alpha})\) in \(c\) maps all occurrences of \(x \in (T \cup V)\) with \(\Upsilon(c,x) = i\) in \(\vec{\alpha}\) to \(\phi(i)\). If the result is defined (i.e., the images of adjacent variables can be concatenated), it is called an instantiated predicate and the result of applying \(\phi\) to all predicates in \(c\), if defined, is called an instantiated clause.

We also introduce range constraint vectors, vectors of pairs of range boundary variables together with a set of constraints on these variables.

**Definition 4.** Let \(V_r = \{r_1, r_2, \ldots\}\) be a set of range boundary variables. A range constraint vector of dimension \(k\) is a pair \((\vec{\rho}, C)\) where \(\vec{\rho} \in (V_r^2)^k\): we define \(V_r(\vec{\rho})\) as the set of range boundary variables occurring in \(\vec{\rho}\). b) \(C\) is a set of constraints \(c_r\) that have one of the following forms:

\[
\begin{align*}
1) & \quad r_1 = r_2, \quad k = r_1, \quad r_1 + k = r_2, \quad k \leq r_1, \quad r_1 \leq k, \quad r_1 \leq r_2 \text{ or } r_1 + k \leq r_2 \quad \text{for } r_1, r_2 \in V_r(\vec{\rho}) \text{ and } k \in \mathbb{N}.
\end{align*}
\]

We say that a range vector \(\phi\) satisfies a range constraint vector \((\rho, C)\) iff \(\phi\) and \(\rho\) are of the same dimension \(k\) and there is a function \(f : V_r \rightarrow \mathbb{N}\) that maps \(\rho(i).l\) to \(\phi(i).l\) and \(\rho(i).r\) to \(\phi(i).r\) for all \(1 \leq i \leq k\) such that all constraints in \(C\) are satisfied. Furthermore, we say that a range constraint vector \((\rho, C)\) is satisfiable iff there exists a range vector \(\phi\) that satisfies it.

**Definition 5.** For every clause \(c\), we define its range constraint vector \((\rho, C)\) w.r.t. a \(w\) with \(|w| = n\) as follows: a) \(\rho\) has dimension \(\Upsilon(c)\) and all range boundary variables in \(\rho\) are pairwise different. b) For all \((r_1, r_2) \in \rho; 0 \leq r_1, r_1 \leq r_2, r_2 \leq n \in C\). For all occurrences \(x\) of terminals in \(c\) with \(i = \Upsilon(c,x)\): \(\rho(i).l+1 = \rho(i).r \in C\). For all \(x, y\) that are variables or occurrences of terminals in \(c\) such that \(xy\) is a substring of one of the arguments in \(c\): \(\rho(\Upsilon(c,x)).r = \rho(\Upsilon(c,y)).l \in C\). These are all constraints in \(C\).

The range constraint vector of a clause \(c\) captures all information about boundaries forming a range, ranges containing only a single terminal, and adjacent variables/terminal occurrences in \(c\).

An RCG derivation consists of rewriting instantiated predicates applying instantiated clauses, i.e., in every derivation step \(\Gamma_1 \Rightarrow \Gamma_2\), we replace the lefthand side of an instantiated clause with its righthand side (w.r.t. a word \(w\)). The language of an RCG \(G\) is the set of strings that can be reduced to the empty word: \(L(G) = \{w \mid S(\langle \epsilon, |w| \rangle) \Rightarrow_{G,w} \epsilon\}\).

The expressive power of RCG lies beyond mild context-sensitivity. As an example, consider the RCG from Fig. 3 that generates a language that is not semilinear.

For simplicity, we assume in the following without loss of generality that empty arguments (\(\epsilon\)) occur only in clauses whose righthand sides are empty.

3 Directional Bottom-Up Chart Parsing

In our directional CYK algorithm, we move a dot through the righthand side of a clause. We therefore have passive items \([A, \phi]\) where \(A\) is a predicate and \(\phi\) a range vector of dimension \(\dim(A)\) and active items. In the latter, while traversing the righthand side of the clause, we keep a record of the left and right boundaries already found for variables and terminal occurrences. This is achieved by subsequently enriching the range constraint vector of the clause. Active items have the form \([A(\vec{x}) \rightarrow \Phi \bullet \Psi, (\rho, C)]\) with \(A(\vec{x}) \rightarrow \Phi \Psi\) a clause, \(\Phi \Psi \neq \epsilon, \Upsilon(A(\vec{x} \rightarrow \Phi \Psi)) = j\) and \((\rho, C)\) a range constraint vector of dimension \(j\). We require that \((\rho, C)\) be satisfiable.\(^3\)

\(^3\)Any RCG can be easily transformed into an RCG satisfying this condition: Introduce a new unary predicate \(Eps\) with a clause \(Eps(\epsilon) \rightarrow \epsilon\). Then, for every clause \(c\) with righthand side not \(\epsilon\), replace every argument \(\epsilon\) that occurs in \(c\) with a new variable \(X\) (each time a distinct one) and add the predicate \(Eps(X)\) to the righthand side of \(c\).

\(^3\)Items that are distinguished from each other only by a bijection of the range variables are considered equivalent. I.e., if the application of a rule yields a new item such that an equivalent one has already been generated, this new one is not added to the set of partial results.
Scan: \[ A(\vec{x}) \rightarrow \varepsilon \in P \] with instantiation \( \psi \)

such that \( \psi(A(\vec{x})) = A(\phi) \)

Initialize: \[ A(\vec{x}) \rightarrow \varepsilon \in P \] with

range constraint vector \((\rho, C), \Phi \neq \varepsilon \)

Complete:

\[ \begin{align*}
A(\vec{x}) & \rightarrow \Phi \bullet B(x_1, \ldots, x_k, y_1, \ldots, y_k) \Phi, (\rho, C) \\
A(\vec{x}) & \rightarrow \Phi B(x_1, \ldots, x_k, y_1, \ldots, y_k) \Phi, (\rho, C)
\end{align*} \]

where \( C' = C \cup \phi_B(j), l = \rho(T(x_j)), l, \phi_B(j), r = \rho(T(y_j)), r \mid 1 \leq j \leq k \).

Convert: \[ \begin{align*}
A(\vec{x}) & \rightarrow \Phi \bullet B(x_1, \ldots, x_k, y_1, \ldots, y_k) \Phi, (\rho, C) \\
A(\vec{x}) & \rightarrow \Phi B(x_1, \ldots, x_k, y_1, \ldots, y_k) \Phi, (\rho, C)
\end{align*} \]

Goal: \( [S, ((0, n_i))] \)

Figure 1: CYK deduction rules

The deduction rules are shown in Fig. 1. The first rule scans the yields of terminating clauses. Initialize introduces clauses with the dot on the left of the right-hand side. Complete moves the dot over a predicate provided a corresponding passive item has been found. Convert turns an active item with the dot at the end into a passive item.

4 The Earley Algorithm

We now add top-down prediction to our algorithm. Active items are as above. Passive items have an additional flag \( p \) or \( c \) depending on whether the item is predicted or completed, i.e., they either have the form \([A, (\rho, C), p]\) where \((\rho, C)\) is a range constraint vector of dimension \( \text{dim}(A) \), or the form \([A, \phi, c]\) where \( \phi \) is a range vector of dimension \( \text{dim}(A) \).

Initialize: \[ [S, ((r_1, r_2)), (0 = r_1, n = r_2), p] \]

Predict-rule: \[ A(x_1, \ldots, x_k, y_1, \ldots, y_k) \rightarrow \Phi \Psi, (\rho, C') \]

where \((\rho', C')\) is obtained from the range constraint vector of the clause \( A(x_1, \ldots, x_k, y_1, \ldots, y_k) \rightarrow \Psi \) by taking all constraints from \( C \), mapping all \( \rho(i), l \) to \( \rho'(T(x_i)), l \) and all \( \rho(i), r \) to \( \rho'(T(y_i)), r \), and then adding the resulting constraints to the range constraint vector of the clause.

Predict-pred:

\[ \begin{align*}
A(...) & \rightarrow \Phi \bullet B(x_1, \ldots, x_k, y_1, \ldots, y_k) \Phi, (\rho, C) \\
A(...) & \rightarrow \Phi B(x_1, \ldots, x_k, y_1, \ldots, y_k) \Phi, (\rho, C)
\end{align*} \]

where \( \rho'(i), l = \rho(T(x_i)), l, \rho'(i), r = \rho(T(y_i)), r \) for all \( 1 \leq i \leq k \) and \( C' = \{c \in C \mid c \text{ contains only range variables from } \rho'\} \).

Scan: \[ \begin{align*}
A(\vec{x}) & \rightarrow \varepsilon \in P \] with an instantiation \( \psi \) satisfying \((\rho, C)\) such that \( \psi(A(\vec{x})) = A(\phi) \)

Figure 2: Earley deduction rules

The deduction rules are listed in Fig. 2. The axiom is the prediction of an \( S \) ranging over the entire input (initialize). We have two predict operations: Predict-rule predicts active items with the dot on the left of the right-hand side, for a given predicted passive item. Predict-pred predicts a passive item for the predicate following the dot in an active item. Scan is applied whenever a predicted predicate can be derived by an \( \varepsilon \)-clause. The rules complete and convert are the ones from the CYK algorithm except that we add flags \( c \) to the passive items occurring in these rules. The goal is again \([S, ((0, n_i)), c]\).

To understand how this algorithm works, consider the example in Fig. 3. The crucial property of this algorithm, in contrast to previous approaches, is the dynamic updating of a set of constraints on range boundaries. We can leave range boundaries unspecified and compute their values in a more incremental fashion instead of guessing all ranges of a clause at once at prediction. 4

For evaluation, we have implemented a directional top-down algorithm where range boundaries are guessed at prediction (this is essentially the algorithm described in Boullier (2000)), and the new Earley-style algorithm. The algorithms were tested on different words of the language \( L = \{a^n \mid n \leq 0\} \). Table 1 shows the number of generated items.

<table>
<thead>
<tr>
<th>Word</th>
<th>Earley</th>
<th>TD</th>
<th>Word</th>
<th>Earley</th>
<th>TD</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a^2)</td>
<td>15</td>
<td>21</td>
<td>(a^{10})</td>
<td>100</td>
<td>539</td>
</tr>
<tr>
<td>(a^4)</td>
<td>30</td>
<td>55</td>
<td>(a^{20})</td>
<td>155</td>
<td>1666</td>
</tr>
<tr>
<td>(a^8)</td>
<td>55</td>
<td>164</td>
<td>(a^{32})</td>
<td>185</td>
<td>1894</td>
</tr>
<tr>
<td>(a^9)</td>
<td>59</td>
<td>199</td>
<td>(a^{64})</td>
<td>350</td>
<td>6969</td>
</tr>
</tbody>
</table>

Table 1: Items generated by both algorithms

Clearly, range boundary constraint propagation increases the amount of information transported in single items and thereby decreases considerably the number of generated items.

5 Conclusion and future work

We have presented a new CYK and Earley parsing algorithms for the full class of RCG. The crucial difference between previously proposed top-down RCG parsers and the new Earley-style algorithm is that while the former compute all clause instantiations during predict operations, the latter

\textsuperscript{4}Of course, the use of constraints makes comparisons between items more complex and more expensive which means that for an efficient implementation, an integer-based representation of the constraints and adequate techniques for constraint solving are required.
Grammar for \(a^n \mid n > 0\): \(S(XY) \rightarrow S(X) \cdot eq(X, Y), S(a_1) \rightarrow \varepsilon, eq(a_1X, a_2Y) \rightarrow eq(X, Y), eq(a_1, a_2) \rightarrow \varepsilon\)

<table>
<thead>
<tr>
<th>Item</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S(((r_1, r_2)), {0 = r_1, r_1 \leq r_2, 2 = r_2}), p)</td>
<td>initialize</td>
</tr>
<tr>
<td>(S(XY) \rightarrow \bullet S(X) \cdot eq(X, Y), {X.I \leq X.r, X.r = Y.l, Y.I \leq Y.r, 0 = X.I, 2 = Y.r})</td>
<td>predict-rule from 1</td>
</tr>
<tr>
<td>(S{((r_1, r_2)), {0 = r_1, r_1 \leq r_2}, p})</td>
<td>predict-pred from 2</td>
</tr>
<tr>
<td>(S, ({0, 1}), c)</td>
<td>scan from 3</td>
</tr>
<tr>
<td>(S(XY) \rightarrow \bullet S(X) \cdot eq(X, Y), {X.I \leq X.r, X.r = Y.Y, Y.I \leq Y.r, 0 = X.I, 1 = X.r})</td>
<td>predict-rule from 3</td>
</tr>
<tr>
<td>(S(XY) \rightarrow S(X) \bullet eq(X, Y), {\ldots, 0 = X.I, 2 = Y.r, 1 = X.r})</td>
<td>complete 2 with 4</td>
</tr>
<tr>
<td>(S(XY) \rightarrow S(X) \bullet eq(X, Y), {X.I \leq X.r, X.r = Y.Y, Y.I \leq Y.r, 0 = X.I, 1 = X.r})</td>
<td>complete 5 with 4</td>
</tr>
<tr>
<td>(eq, {(r_1, r_2), {r_3, r_3 \leq r_4, 0 = r_1, 2 = r_4, 1 = r_2}})</td>
<td>predict-pred from 6</td>
</tr>
<tr>
<td>(eq(a_1X, a_2Y) \rightarrow \bullet eq(X, Y), {a_1 + 1 = a_1, a_1.X, a_1.r = X.I, X.I \leq X.r, a_2.r = Y.I, Y.I \leq Y.r, X.r = a_2.r, 0 = a_1, 1 = X.r, 2 = Y.r})</td>
<td>predict-rule from 8</td>
</tr>
<tr>
<td>(eq, ({0, 1}, (1, 2)), c)</td>
<td>scan 8</td>
</tr>
<tr>
<td>(S(XY) \rightarrow S(X) \cdot eq(X, Y), {\ldots, 0 = X.I, 2 = Y.r, 1 = X.r, 1 = Y.I})</td>
<td>complete 6 with 10</td>
</tr>
<tr>
<td>(S, ((0, 2)), c)</td>
<td>convert 11</td>
</tr>
</tbody>
</table>

Figure 3: Trace of a sample Earley parse

avoids this using a technique of dynamic updating of a set of constraints on range boundaries. Experiments show that this significantly decreases the number of generated items, which confirms that range boundary constraint propagation is a viable method for a lazy computation of ranges.

The Earley parser could be improved by allowing to process the predicates of the righthand sides of clauses in any order, not necessarily from left to right. This way, one could process predicates whose range boundaries are better known first. We plan to include this strategy in future work.

References


