Parsing Linear Context-Free Rewriting Systems

October 10th 2005,
IWPT'05 Vancouver

Håkan Burden and Peter Ljunglöf
Department of Linguistics, Göteborg University

Abstract

We present a description of four algorithms for parsing Linear Context-Free Rewriting Systems. The algorithms are described as deductive parsing systems, in the spirit of Shieber, Schabes & Pereira (1995).

Motivation

Most of the existing parsing algorithms for Linear Context-Free Rewriting Systems (LCFRS; Vijay-Shanker et al., 1987) are designed for theoretical purposes:

An important subclass of Grammatical Framework (Ranta, 2004) is equivalent to LCFRS (Ljunglöf, 2004).

Minimalist Grammars (Stabler, 1997) can be parsed as LCFRS (Michaelis, 1998).

Linear Context-Free Rewriting Systems

An LCFRS is a linear, non-erasing Multiple Context-Free Grammar (MCFG; Seki et al., 1991).

We write a combined MCFG rule in the following way,

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

For convenience, we use records instead of tuples.

An example LCFRS grammar for cross serial dependencies

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

The grammar generates sentences on the form

a b a c d c

i.e: bd , $abcd$ and $aacc$ are recognized but not abc nor $abcdabcd$.

Swiss german cross dependencies

A popular example from Shieber (1985) for swiss german:

...mer em Hans es huus hälfed aastriiche

Parsing LCFRS

Until now there were three ways of parsing LCFRS:

- CKY-variant (Seki et al, 1991)

- Boolean matrix multiplication (Nakanishi et al., 1997)

- Earley-variant (Albro, 2002)

Our contribution

We give four new algorithms:

- Naive,
- Approximative,
- Active and
- Incremental.

Ranges

We borrow the idea of ranges from Boullier (2000).

In the string w a range ρ is a pair of indices, (i, j), s.t. $0 \le i \le j \le |w|$.
The range (i, j) then denotes the substring $w_{i+1} \dots w_j$.

If $i = j$ the range is empty and denotes the empty string.

Concatenation of two ranges is non-deterministic,

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

Range restriction

To get the ranges of a substring s in a sentence w we define range restriction of s with respect to w as

$$\langle s \rangle = \Big\{ (i, j\,) \mid s = w_{i+1} \dots w_j \Big\}$$

Range restriction of a linearization record, Φ , is written $\langle \Phi \rangle$. Range restriction fails if range concatenation fails for two adjacent ranges. Any argument projections, $A_i.p$, are left unaffected.

Examples

Given the string abb we get

$$\begin{aligned}
\langle a\rangle &= \{(0,1)\} \\
\langle b\rangle &= \{(1,2),(2,3)\} \\
\langle a\ b\rangle &= \langle a\rangle \cdot \langle b\rangle = \{(0,2)\}
\end{aligned}$$

and

$$\langle A.p\ \ a\ \ b\ \ B.q\rangle = \{A.p\ \ (0,2)\ B.q\}$$

Parsing as deduction (Schieber et al., 1995):

In general we have:

$$\frac{\gamma_1 \cdots \gamma_n}{\gamma}\{\mathcal{C}$$

The standard inference rule Combine might look like this for CFG:

$$\frac{\begin{array}{l}[S \rightarrow NP \bullet VP; \rho'] \\ [VP; \rho'']\end{array}}{[S \rightarrow NP \quad VP\bullet; \rho]}\ \{\ \rho \in \rho' \cdot \rho''$$

The Naive algorithm

Passive item $[A; \Gamma]$
Active item for the rule $A \to f\,[B_1 \ldots B_\delta] := \Psi$
has the form

$$[A \to f\,[B_1 \ldots B_j \bullet B_{j+1} \ldots B_\delta]; \Phi; \Gamma_1 \ldots \Gamma_j]$$

where $\Phi = \Psi[B_1/\Gamma_1 \ldots B_\delta/\Gamma_\delta]$

Inference rules: Predict, Combine and Convert

Naive Predict

$$\frac{}{[A \to f[\bullet B_1 \dots B_\delta]; \Phi; \quad]} \begin{cases} A \to f[B_1 \dots B_\delta] := \Psi \\ \Phi \in \langle \Psi \rangle \end{cases}$$

Predict an active item for every grammar rule.

Naive Combine

$$\frac{\begin{array}{l}[A \to f[B_1 \ldots B_{k-1} \bullet B_k \quad B_{k+1} \ldots B_\delta]; \Psi; \Gamma_1 \ldots \Gamma_{k-1}] \\ [B_k; \Gamma_k]\end{array}}{[A \to f[B_1 \ldots B_{k-1} B_k \bullet B_{k+1} \ldots B_\delta]; \Phi; \Gamma_1 \ldots \Gamma_{k-1}, \Gamma_k]} \{ \Phi \in \Psi[B_k/\Gamma_k]$$

An active item searching for B_k can be combined with a passive item that has found B_k .

Naive Convert

$$\frac{[A \rightarrow f[B_1 \ldots B_\delta \bullet]; \Phi; \Gamma_1 \ldots \Gamma_\delta]}{[A; \Gamma]} \left\{ \Gamma \equiv \Phi \right.$$

A fully instantiated active item is converted to a passive item.

The Approximative algorithm

A variant of the Naive algorithm.

We use Context-Free approximation instead of range-restriction:

- Parse the sentence using a Context-Free approximation

- Recover the resulting chart into a LCFRS chart

The Active algorithm

Passive item $[A; \Gamma]$.
Active item for the rule $A \to f\,[B_1 \ldots B_\delta] := \{\Phi; r = \alpha\beta; \Psi\}$

has the form

$$[A \to f\,[B_1 \ldots B_\delta]; \Gamma, r = \rho \bullet \beta, \Psi; \Gamma_1 \ldots \Gamma_\delta]$$

Inference rules: Predict, Complete, Scan, Combine and Convert.

The epsilon-range

We use ρ^ϵ to simultaneously denote all empty ranges (i, i).

Range restricting the empty string gives $\langle\epsilon\rangle = \rho^\epsilon$.

Concatenation: $\rho \cdot \rho^\epsilon = \rho^\epsilon \cdot \rho = \rho$.

Active Predict

$$\frac{}{[A \to f\ [B_1 \ldots B_\delta]; \quad , r = \rho^\epsilon \bullet \alpha, \Phi; \Gamma_1 \ldots \Gamma_\delta]} \{A \to f\ [B_1 \ldots B_\delta] := \{r = \alpha; \Phi\}$$

Predict an active item that has found the empty range for every rule in the grammar.

Active Complete

$$\frac{[A \rightarrow f\ [B_1 \ldots B_\delta]; \Gamma, r = \rho \bullet \epsilon, q = \alpha; \Phi; \Gamma_1 \ldots \Gamma_\delta]}{[A \rightarrow f\ [B_1 \ldots B_\delta]; \Gamma; r = \rho, q = \rho^\epsilon \bullet \alpha, \Phi; \Gamma_1 \ldots \Gamma_\delta]}$$

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

Active Scan

$$\frac{[A \to f\ [B_1 \ldots B_\delta]; \Gamma, r = \rho \bullet s\alpha, \Phi; \Gamma_1 \ldots \Gamma_\delta]}{[A \to f\ [B_1 \ldots B_\delta]; \Gamma, r = \rho' \bullet \alpha, \Phi; \Gamma_1 \ldots \Gamma_\delta]} \left\{ \rho' \in \rho \cdot \langle s \rangle \right.$$

Scanning is applied when the next symbol is a terminal.

Active Combine

$$\frac{\begin{array}{l}[A \to f\ [B_1 \ldots B_\delta]; \Gamma, r = \rho \bullet B_i.q\,\alpha, \Phi; \Gamma_1 \ldots \Gamma_i \ldots \Gamma_\delta] \\ [B_i; \Gamma']\end{array}}{[A \to f\ [B_1 \ldots B_\delta]; \Gamma, r = \rho' \bullet \alpha, \Phi; \Gamma_1 \ldots \Gamma' \ldots \Gamma_\delta]} \left\{ \begin{array}{l} \rho' \in \rho \cdot \Gamma'.q \\ \Gamma_i \subseteq \Gamma' \end{array} \right.$$

A passive item, with B_i as its category, can be combined with an active item searching for a projection B_i .

Active Convert

$$\frac{[A \rightarrow f\ [B_1 \ldots B_\delta]; \Gamma, r = \rho \bullet \epsilon \quad ; \Gamma_1 \ldots \Gamma_\delta]}{[A; \Gamma, r = \rho]}$$

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

The Incremental algorithm

A variant of the Active algorithm.

The linearization records are treated as sets, not sequences.

- We cannot use the ρ^ϵ , so we get more items.

- Fewer matches when applying Combine.

Prediction strategies

Predict an item for the rule $A \to f\,[\vec{B}]$ with the linearization row $r = \alpha$ if ...

- ... there is an item looking for $A.r$ (Top-down) or
- ... there is a passive item that has found the first symbol in α (Bottom-up).

A small comparison of runtimes

Parsing three sentences with a non-trivial grammar of 561 rules gives the following table:

Sentence	Albro's algorithm	Active bottom-up
The boy is young	1.1 s	0.2 s
The boy is so young	2.0 s	0.3 s
They had forgotten that the boy who told the story is so young	828 s	46 s

The grammar is called 'Larsonian' and automatically generated from a Minimalist grammar

TODO

- Extensive evaluation.
- Filtering techniques.