Parsing beyond context-free grammar: Range Concatenation Grammar Parsing

Laura Kallmeyer, Wolfgang Maier

Sommersemester 2009

Overview

Motivation (1)

Why are Range Concatenation Grammars (RCG) of interest?

Expressive power of RCGs: RCGs

- generate all languages in PTIME (Bertsch & Nederhof, 2001);
- can deal with permutations as occurring in scrambling phenomena in free word order languages (Boullier, 1999);
- can deal with non-semilinear natural language phenomena such as Chinese number names (Radzinski, 1991, Boullier, 1999).

Motivation (2)

Application aspects of RCG: RCGs

- allow to describe different linguistic aspects of strings in a single system (Sagot, 2005);
- can be used as a pivot formalism for parsing (Kallmeyer et al., 2008);
- allow an intuitive description of discontinuities in natural language and are therefore well suited for grammar extraction from treebanks with crossing branches (Maier & Søgaard, 2008, Kuhlmann & Satta, 2009)
- can be used for machine translation (Søgaard, 2008).

RCG (1)

Idea: The productions of RCGs (called *clauses*) rewrite predicates ranging over parts of the input by other predicates.

Ex.: $S(aXb) \rightarrow S(X)$ signifies that S holds for a part of the input if

- this part starts with an a and ends with a b and
- S also holds for the part between the a and the b.

The RCG with clauses $S(aXb) \rightarrow S(X), S(c) \rightarrow \epsilon$ generates the language $\{a^n c b^n \mid n \geq 0\}$.

RCG (2)

A RCG is a tuple $G = \langle N, T, V, S, P \rangle$ such that

- N is a finite set of predicates, each with a fixed arity, $S \in N$ is a unary start predicate;
- T and V are disjoint alphabets of terminals and variables,
- P is a finite set of clauses of the form

$$\psi_0 \rightarrow \psi_1 \ldots \psi_m, m \geq 0$$

  where each of the $\psi_i, 0 \leq i \leq m$, is a predicate of the form $A_i(\alpha_1, \ldots, \alpha_{dim(A_i)})$ with $A_i \in N$ and $\alpha_j \in (T \cup V)^*$ for $1 \leq j \leq dim(A_i)$.

RCG (3)

For a given clause, an instantiation with respect to a string $w = t_1 \ldots t_n$ maps all variables and all occurrences of terminals in the clause to ranges $\langle i, j \rangle \mid 0 \leq i \leq j \leq |w|\}$ such that

- all occurrences of a terminal t are mapped to a range whose yield is a t, and
- adjacent variables/occurrences of terminals in one of the arguments are mapped on adjacent ranges, i.e., ranges $\langle i, j \rangle, \langle k, l \rangle$ with $j = k$.

A derivation step consists of replacing the lhs of an instantiated clause with its rhs.

The language of an RCG G is the set of strings that can be reduced to the empty word:

$L(G) = \{w \mid S(\langle 0, |w| \rangle) \stackrel{*}{\Rightarrow} \epsilon$ with respect to $w\}$.

RCG (4)

Example: The RCG with clauses

$S(XY) \rightarrow S(X)eq(X, Y)$

$S(a) \rightarrow \epsilon$

$eq(aX, aY) \rightarrow eq(X, Y)$

$eq(a, a) \rightarrow \epsilon$

generates the language $\{a^{2^n} \mid n \geq 0\}$

RCG (5)

As an example consider the reduction of $w = aaaa$:

$$\begin{array}{cccccc} S(X & Y) & \rightarrow & S(X) & eq(X, & Y) \\ \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\ \langle 0,2\rangle & \langle 2,4\rangle & & \langle 0,2\rangle & \langle 0,2\rangle & \langle 2,4\rangle \\ aa & aa & & aa & aa & aa \end{array}$$

With this instantiation,

$S(\langle 0,4\rangle) \Rightarrow S(\langle 0,2\rangle)eq(\langle 0,2\rangle, \langle 2,4\rangle)$.

RCG (6)

$$\begin{array}{cccccc} S(X & Y) & \rightarrow & S(X) & eq(X, & Y) \\ \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\ \langle 0,1\rangle & \langle 1,2\rangle & & \langle 0,1\rangle & \langle 0,1\rangle & \langle 1,2\rangle \\ a & a & & a & a & a \end{array}$$

$$\begin{array}{ccc} S(a) & \rightarrow & \epsilon \\ \downarrow & & \\ \langle 0,1\rangle & & \\ a & & \end{array} \qquad \begin{array}{cccc} eq(a, & a) & \rightarrow & \epsilon \\ \downarrow & \downarrow & & \\ \langle 0,1\rangle & \langle 1,2\rangle & & \\ a & a & & \end{array}$$

leads to $S(\langle 0,2\rangle) \Rightarrow S(\langle 0,1\rangle)eq(\langle 0,1\rangle, \langle 1,2\rangle) \overset{*}{\Rightarrow} \epsilon$

RCG (7)

$$\begin{array}{cccccc} eq(a & X & a & Y) & \rightarrow & eq(X, & Y) \\ \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow \\ \langle 0,1\rangle & \langle 1,2\rangle & \langle 2,3\rangle & \langle 3,4\rangle & & \langle 1,2\rangle & \langle 3,4\rangle \\ a & a & a & a & & a & a \end{array}$$

$$\begin{array}{cccc} eq(a, & a) & \rightarrow & \epsilon \\ \downarrow & \downarrow & & \\ \langle 1,2\rangle & \langle 3,4\rangle & & \\ a & a & & \end{array}$$

leads to $eq(\langle 0,2\rangle, \rangle 2,4\langle) \Rightarrow eq(\langle 1,2\rangle, \langle 3,4\rangle) \Rightarrow \epsilon$

Top-Down Parsing (1)

Non-directional top-down parsing:

Idea: instantiate the start predicate with the entire string and check recursively if there is a way to reduce all righthand side predicates to ϵ.

Item form: $[A, \phi, \mathit{flag}]$, where

- A is a predicate,
- ϕ is a range vector of dimension $dim(A)$ and
- $\mathit{flag} \in \{c, p\}$ indicates if the item has been completed or predicted.

Top-Down Parsing (2)

Axiom: $\dfrac{}{[S, (\langle 0, n\rangle), p]}$

The **predict** operation predicts new items for previously predicted items.

Predict: $\dfrac{[A_0, \phi, p]}{[A_1, \phi_1, p] \dots [A_k, \phi_k, p]}$

where there is an instantiated clause $A_0(\phi) \rightarrow A_1(\phi_1) \dots A_k(\phi_k)$.

Top-Down Parsing (3)

Complete changes the flag on a lefthand side predicate if the righthand side is completed.

Complete: $\dfrac{[A_0, \phi, p], [A_1, \phi_1, c] \dots [A_k, \phi_k, c]}{[A_0, \phi, c]}$

where there is an instantiated clause $A_0(\phi) \rightarrow A_1(\phi_1) \dots A_k(\phi_k)$.

(ϵ-clauses are covered as a special case of *complete*.)

Goal: $[S, (\langle 0, n\rangle), c]$.

Top-Down Parsing (4)

Directional Top-Down Parsing: The righthand sides of a predicted clause are completed one after the other (Boullier, 2000).

We use

1. passive items as before, and
2. Active items $[A(\vec{x}) \rightarrow \Phi \bullet \Psi, \phi]$ where
   - $A(\vec{x}) \rightarrow \Phi\Psi$ is a clause, and
   - ϕ is a range vector of dimension $j = \Upsilon(A(\vec{x}) \rightarrow \Phi\Psi)$ that gives an instantiation of the clause

$\Upsilon(c)$ gives the number of variables and occurrences of terminals in the clause c.

Top-Down Parsing (5)

The axiom is the same as before.

Predict-rule predicts active items with the dot on the left of the righthand side, for a given predicted passive item.

Predict-rule: $\dfrac{[A, \psi, p]}{[A(\vec{x}) \rightarrow \bullet\Psi, \phi]} \quad \phi(A(\vec{x})) = A(\psi)$

Predict-pred predicts a passive item for the predicate following the dot in an active item:

Predict-pred: $\dfrac{[A(\vec{x}) \rightarrow \Phi \bullet B(\vec{y})\Psi, \phi]}{[B, \psi, p]} \quad \phi(B(\vec{y})) = B(\psi)$

Top-Down Parsing (6)

The **scan** operation switches the flag on a item describing a predicted predicate to completed, given that there is a corresponding ϵ-clause:

Scan: $\dfrac{[A, \phi, p]}{[A, \phi, c]}$ there is a clause instantiation $A(\phi) \rightarrow \epsilon$

Complete moves the dot over a predicate in the righthand side of an active item if the corresponding passive item has been completed.

Complete: $\dfrac{[B, \phi_B, c], [A(\vec{x}) \rightarrow \Phi \bullet B(\vec{y})\Psi, \phi]}{[A(\vec{x}) \rightarrow \Phi B(\vec{y}) \bullet \Psi, \phi]}$ $\phi(B(\vec{y})) = B(\phi_B)$

Top-Down Parsing (7)

Once the dot has reached the right end of a clause, we can **convert** the active item into a passive item:

Convert: $\dfrac{[A(\vec{x}) \rightarrow \Phi\bullet, \phi]}{[A, \psi, c]}$ $\phi(A(\vec{x})) = A(\psi)$

Goal (as before): $[S, (\langle 0, n\rangle), c]$.

Obvious problem: **Predict-rule** has to compute all possible instantiations of A-clauses, given an instantiated A-predicate.

Bottom-Up Parsing (1)

CYK (non-directional bottom-up): Items $[A, \phi]$ where A a predicate, ϕ a range vector of dimension $dim(A)$.

Scan: $\dfrac{}{[A, \phi]}$ there is a clause instantiation $A(\phi) \rightarrow \epsilon$

Complete: $\dfrac{[A_1, \phi_1] \ldots [A_k, \phi_k]}{[A, \phi]}$

where $A(\phi) \rightarrow A_1(\phi_1) \ldots A_k(\phi_k)$ is an instantiated clause.

Goal item: $[S, (\langle 0, n\rangle)]$.

Bottom-Up Parsing (2)

Directional Bottom-Up Parsing: We move a dot through the righthand side of a clause.

This means that for active items, before having reached the end of a righthand side, we don't know all the range boundaries involved.

Therefore, instead of range vectors, we have range constraint vectors in the active items.

Bottom-Up Parsing (3)

A range constraint vector is a vector of pairs $\langle r_l, r_r \rangle$ of range boundary variables together with a set of constraints on all variables in the vector.

The constraints can have the following forms:

$r_1 = r_2$, $k = r_1$, $r_1 + k = r_2$,

$k \leq r_1$, $r_1 \leq k$, $r_1 \leq r_2$ or $r_1 + k \leq r_2$

for range boundary variables r_1, r_2 and $k \in \mathbb{N}$.

Example:

The clause $S(XY) \rightarrow S(X)eq(X,Y)$ comes with the range constraint vector $\langle(\langle X.l, X.r\rangle), (\langle Y.l, Y.r\rangle), \{0 \leq X.l,\ X.l \leq X.r,\ X.r = Y.l,\ Y.l \leq Y.r,\ Y.r \leq n\}\rangle$.

Bottom-Up Parsing (4)

Passive items $[A, \phi]$ as in non-directional CYK.

Active items $[A(\vec{x}) \rightarrow \Phi \bullet \Psi, \langle \rho, C \rangle]$ with

- $A(\vec{x}) \rightarrow \Phi\Psi$ a clause, $\Phi\Psi \neq \epsilon$,
- $\langle \rho, C \rangle$ a range constraint vector of dimension j where $\Upsilon(A(\vec{x} \rightarrow \Phi\Psi)) = j$. We require that $\langle \rho, C \rangle$ be satisfiable.

The **goal** is again $[S, (\langle 0, n \rangle)]$.

Bottom-Up Parsing (5)

The **scan** rule is the same as in the basic algorithm.

Initialize introduces clauses with the dot on the left of the righthand side:

$$\frac{}{[A(\vec{x}) \rightarrow \bullet\Phi, \langle \rho, C \rangle]} \quad A(\vec{x}) \rightarrow \Phi \text{ a clause with range constraint vector } \langle \rho, C \rangle, \Phi \neq \epsilon$$

Bottom-Up Parsing (6)

Complete rule moves the dot over a predicate in the righthand side of an active item:

$$\frac{[B, \phi_B], [A(\vec{x}) \rightarrow \Phi \bullet B(x_1 \ldots y_1, \ldots, x_k \ldots y_k)\Psi, \langle \rho, C \rangle]}{[A(\vec{x}) \rightarrow \Phi B(x_1 \ldots y_1, \ldots, x_k \ldots y_k) \bullet \Psi, \langle \rho, C' \rangle]}$$

where

$C' = C \cup \{\phi_B(j).l = \rho(\Upsilon(x_j)).l,\ \phi_B(j).r = \rho(\Upsilon(y_j)).r \mid 1 \leq j \leq k\}$.

$\Upsilon(x)$ gives the index of element x in the clause.

Bottom-Up Parsing (7)

Convert turns an active item with the dot at the end into a passive item:

$$\text{Convert:}\quad \frac{[A(\vec{x}) \rightarrow \Psi\bullet, \langle\rho, C\rangle]}{[A, \phi]}$$

where there is an instantiation ψ of $A(\vec{x}) \rightarrow \Psi$ that satisfies $\langle\rho, C\rangle$ such that $\psi(A(\vec{x})) = A(\phi)$.

Earley Parsing with Constraint Propagation (1)

Idea: add **predict** to the directional bottom-up parser.

Passive items:

- $[A, \langle\rho, C\rangle, p]$ where $\langle\rho, C\rangle$ is a range constraint vector of dimension $dim(A)$ (for a predicted item)
- or $[A, \phi, c]$ for completed items where ϕ is a range vector of dimension $dim(A)$.

The active items are the same as in the directional bottom-up case.

Earley Parsing with Constraint Propagation (2)

The axiom is the prediction of an S ranging over the entire input:

$$\text{Initialize:}\quad \frac{}{[S, \langle(\langle r_1, r_2\rangle), \{0 = r_1, n = r_2\}\rangle, p]}$$

Predict-rule predicts active items with the dot on the left of the righthand side, for a given predicted passive item:

$$\text{Predict-rule:}\quad \frac{[A, \langle\rho, C\rangle, p]}{[A(x_1 \ldots y_1, \ldots, x_k \ldots y_k) \rightarrow \bullet\Psi, \langle\rho', C'\rangle]}$$

where $\langle\rho', C'\rangle$ is obtained from the range constraint vector of the clause $A(x_1 \ldots y_1, \ldots, x_k \ldots y_k) \rightarrow \Psi$ by taking all constraints from C, mapping all $\rho(i).l$ to $\rho'(\Upsilon(x_i)).l$ and all $\rho(i).r$ to $\rho'(\Upsilon(y_i)).r$, and then adding the resulting constraints to the range constraint vector of the clause.

Earley Parsing with Constraint Propagation (3)

Predict-pred predicts a passive item for the predicate following the dot in an active item:

$$\text{Predict-pred:}\quad \frac{[A(\ldots) \rightarrow \Phi \bullet B(x_1 \ldots y_1, \ldots, x_k \ldots y_k)\Psi, \langle\rho, C\rangle]}{[B, \langle\rho', C'\rangle, p]}$$

where $\rho'(i).l = \rho(\Upsilon(x_i)).l$, $\rho'(i).r = \rho(\Upsilon(y_i)).r$ for all $1 \leq i \leq k$ and $C' = \{c \mid c \in C, c$ contains only range variables from $\rho'\}$.

Scan applies whenever a predicted predicate can be derived by an ϵ-clause:

$$\text{Scan:}\quad \frac{[A, \langle\rho, C\rangle, p]}{[A, \phi, c]}$$

where there is a clause $A(\vec{x}) \rightarrow \epsilon$ with a possible instantiation ψ that satisfies $\langle\rho, C\rangle$ such that $\psi(A(\vec{x})) = A(\phi)$.

Earley Parsing with Constraint Propagation (4)

Finally, deduction rules for **complete** and **convert** are the ones from the directional bottom-up algorithm except that we add flags c to the passive items occurring in these rules.

Again, the **goal item** is $[S, (\langle 0, n\rangle), c]$.

Earley Parsing with Constraint Propagation (5)

Evaluation: Comparison between the numbers of items generated with the directional top-down and the Earley-style algorithm:

RCG for a^{2^n}

word	Earley	top-down
a^2	15	21
a^4	30	55
a^8	55	164
a^9	59	199

word	Earley	top-down
a^{16}	100	539
a^{30}	155	1666
a^{32}	185	1894
a^{64}	350	6969

Conclusion

- We have seen different parsing strategies for RCG formulated with deduction rules.
- The use of deduction rules allows a clean comparison of the various algorithms.
- Crucial difference between directional top-down algorithm and new Earley-style algorithm: range boundary constraint propagation in the latter allows for a lazy computation of clause instantiations.
- As a consequence, the number of items generated decreases considerably.