Overview
1. Motivation
2. Range Concatenation Grammars
3. Top-Down Parsing
4. Bottom-Up Parsing
5. Earley Parsing with Constraint Propagation
6. Conclusion
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 b^n | 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 \( (i, j) | 0 \leq i \leq j \leq |w| \) 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
  \( (i, j), (k, l) \) 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:
\( L(G) = \{ w | S(\langle 0, |w| \rangle) \Rightarrow \epsilon \ \text{with respect to} \ w \} \).

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} | n \geq 0 \}\}.
As an example consider the reduction of $w = aaaa$:

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

$$\langle 0, 2 \rangle \langle 2, 4 \rangle \langle 0, 2 \rangle \langle 2, 4 \rangle$$

With this instantiation,

$$S((0, 4)) \Rightarrow S((0, 2))_{eq((0, 2), (2, 4))}.$$
Top-Down Parsing (2)

Axiom: $[S, ((0, n)), p]$  

The predict operation predicts new items for previously predicted items.  

Predict: $[A_0, \phi, p] \rightarrow [A_1, \phi_1, p] \ldots [A_k, \phi_k, p]$  

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

Top-Down Parsing (3)

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

Complete: $[A_0, \phi, p], [A_1, \phi_1, c] \ldots [A_k, \phi_k, c]$  

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

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

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

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  
   • $\phi$ 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: $[A(\vec{x}) \rightarrow \Phi \bullet \Psi, \phi]$  

$\phi(A(\vec{x})) = A(\psi)$  

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

Predict-pred: $[B(\vec{y}) \rightarrow \Phi \bullet B(\vec{y}), \phi]$  

$\phi(B(\vec{y})) = B(\psi)$
Top-Down Parsing (6)

The \textit{scan} operation switches the flag on a item describing a predicted predicate to completed, given that there is a corresponding $\epsilon$-clause:

\begin{align*}
\text{Scan: } & [A, \phi, p] \rightarrow [A, \phi, c] \\
& \text{there is a clause instantiation } A(\phi) \rightarrow \epsilon
\end{align*}

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

\begin{align*}
\text{Complete: } & [B, \phi_B, c], [A(\vec{x}) \rightarrow \Phi \bullet B(\vec{y}) \Psi, \phi] \\
& \rightarrow [A(\vec{x}) \rightarrow \Phi B(\vec{y}) \bullet \Psi, \phi] \\
& \phi(B(\vec{y})) = B(\phi_B)
\end{align*}

Bottom-Up Parsing (1)

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

\begin{align*}
\text{Scan: } & [A, \phi] \\
& \text{there is a clause instantiation } A(\phi) \rightarrow \epsilon
\end{align*}

\begin{align*}
\text{Complete: } & [A_1, \phi_1] \ldots [A_k, \phi_k] \\
& [A, \phi] \\
& \text{where } A(\phi) \rightarrow A_1(\phi_1) \ldots A_k(\phi_k) \text{ is an instantiated clause.}
\end{align*}

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

Top-Down Parsing (7)

Once the dot has reached the right end of a clause, we can \textit{convert} the active item into a passive item:

\begin{align*}
\text{Convert: } & [A(\vec{x}) \rightarrow \Phi \bullet, \phi] \\
& \rightarrow [A, \psi, c] \\
& \phi(A(\vec{x})) = A(\psi)
\end{align*}

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

\textbf{Obvious problem: \textbf{Predict-rule}} has to compute all possible instantiations of $A$-clauses, given an instantiated $A$-predicate.
A range constraint vector is a vector of pairs \( (r_l, r_r) \) of range boundary variables together with a set of constraints on all variables in the vector.

The constraints can have the following forms:

- \( r_1 = r_2 \)
- \( k = r_1 \)
- \( r_1 + k = r_2 \)
- \( k \leq r_1 \)
- \( r_1 \leq k \)
- \( r_1 \leq r_2 \)
- \( r_1 + k \leq r_2 \)

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 \( (\langle X.l, X.r \rangle, \langle Y.l, Y.r \rangle, \{ 0 \leq X.l, X.l \leq X.r, Y.l \leq Y.r, Y.r \leq n \}) \).
Bottom-Up Parsing (7)

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

**Convert:**

\[
\begin{aligned}
\text{Convert:} & \quad [A(\vec{x}) \rightarrow \Psi \bullet (\rho, C)] \\
& \quad \rightarrow [A, \phi]
\end{aligned}
\]

where there is an instantiation \(\psi\) of \(A(\vec{x}) \rightarrow \Psi\) that satisfies \((\rho, C)\) 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:

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

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:

**Initialize:**

\[
[S, ((r_1, r_2), \{0 = r_1, n = r_2\}), p]
\]

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

**Predict-rule:**

\[
\begin{aligned}
\text{Predict-rule:} & \quad [A, (\rho, C), p] \\
& \quad \rightarrow [A(x_1 \ldots y_1, \ldots x_k \ldots y_k) \rightarrow \Psi, (\rho', C')] \\
& \text{where } (\rho', C') \text{ is obtained from the range constraint vector of the clause } A(x_1 \ldots y_1, \ldots x_k \ldots y_k) \rightarrow \Psi \text{ by taking all constraints from } C, \text{ mapping all } \rho(i).l \text{ to } \rho'(T(x_i)).l \text{ and all } \rho(i).r \text{ to } \rho'(T(y_i)).r, \text{ and then adding the resulting constraints to the range constraint vector of the clause.}
\end{aligned}
\]

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

**Scan:**

\[
\begin{aligned}
\text{Scan:} & \quad [A, (\rho, C), p] \\
& \quad \rightarrow [A, \phi, c]
\end{aligned}
\]

where there is a clause \(A(\vec{x}) \rightarrow \epsilon\) with a possible instantiation \(\psi\) that satisfies \((\rho, C)\) 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 $[S, ((0, n)), c]$. 

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

<table>
<thead>
<tr>
<th>word</th>
<th>Earley</th>
<th>top-down</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a^2$</td>
<td>15</td>
<td>21</td>
</tr>
<tr>
<td>$a^4$</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>$a^8$</td>
<td>55</td>
<td>164</td>
</tr>
<tr>
<td>$a^9$</td>
<td>59</td>
<td>199</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>Earley</th>
<th>top-down</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a^{16}$</td>
<td>100</td>
<td>539</td>
</tr>
<tr>
<td>$a^{30}$</td>
<td>155</td>
<td>1666</td>
</tr>
<tr>
<td>$a^{32}$</td>
<td>185</td>
<td>1894</td>
</tr>
<tr>
<td>$a^{64}$</td>
<td>350</td>
<td>6969</td>
</tr>
</tbody>
</table>

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.