Introduction (1)

- LR parsing: Left-to-right scanning and Right-to-left reduction
- We compile a finite-state automaton from the grammar (offline) and use it to guide actions during parsing (online)
- What does the automaton represent?
  - States: Correspond to sets of items closed under prediction
  - Edges: Correspond to scanning a terminal symbol or consuming an already recognized nonterminal

Roughly, LR parsing is Earley parsing with precompiled predictions.

Introduction (2)

An LR automaton is typically represented by two tables.

- The Action table lists what action must be performed (shift or reduce). This action depends on
  - the current state in the automaton
  - the next preterminal to be read
- The Goto table lists the states where the automaton has to go after reducing a production

Overview

1. Introduction
2. Construction of the automaton
3. The recognizer
Introduction (3)
In CFG LR parsing, we dispose of two operations on a stack:

1. *shift*(k): Scans a terminal, pushes the corresponding pre-terminal on the stack and switches to state k

2. *reduce*(A): The RHS of some production $A \rightarrow A_1 \ldots A_n$ has been recognized, i.e. is on the stack. *reduce*(A) pops $A_1, \ldots, A_n$ from the stack and pushes the LHS $A$ on the stack, then switches to the next state (provided by the *goto* table)

Introduction (4)

- Nederhof (1998) extends traditional LR parsing to TAG
- His algorithm is based on
  - a LR parse automaton (automaton)
  - a function to scan the next symbol of the input: *shift*(Δ, aw)
  - two functions to reduce partial results on the stack: *reduce*:Subtree(Δ, w) and *reduce*:Auxtree(Δ, w)
  where w is the input and Δ is the LR stack.

- Nederhof (1998) mentions an implementation of the parser generator
- LR automaton generation for the XTAG grammar seemed to be feasible

Introduction (5)
Notations:
- $N(t)$ is the set of nodes of a tree t.
- children($N$) is the list of the children of a node $N$, given in linear precedence order.

Introduction (6)
- The elementary trees are extended with artificial new nodes:
  - For each $t \in I \cup A$, we add a unique node $\top$ immediately dominating $R_t$ (the root of $t$).
  - For each $t \in A$, we add a unique node $\bot$ immediately dominated by $F_t$ (the foot of $t$).

- For a $t \in I \cup A$, $(t, N)$ denotes the subtree of $t$ rooted in $N$.
  $T = I \cup A \cup \{(t, N) | t \in I \cup A, N \in N(t)\}$ is the set of all subtrees of elementary trees, including the elementary trees themselves.

Assume that our TAG has no substitution nodes and does not contain empty words.
Construction of the automaton (1)

- The states of the LR automaton are sets of items
- Transitions are labeled with terminals and nonterminals

An item represents a subtree of height 1 (mother node \( N \) and its daughters) in one of the \( \tau \in T \) together with a dot • that specifies up to which daughter the subtree has been recognized. This subtree is notated as a dotted production \( N \rightarrow \alpha \bullet \beta \).

Construction of the automaton (2)

Items have the form \([\tau, N \rightarrow \alpha \bullet \beta]\), where
- \( \tau \in T \),
- \( N \in N(\tau) \), and
- \( \alpha \beta \) are the daughters of \( N \).

An item is called **completed** if it has the form
- either \([t, \top \rightarrow R] \) with \( t \in I \cup A \),
- or \([([t, N], N \rightarrow \alpha])\).

Definition of the closure of a state \( q \):

Let \( q \) be a set of items. 

\( \text{closure}(q) \) is then defined by the following inference rules:

\[
\begin{align*}
\frac{x \in q}{[\tau, N \rightarrow \alpha \bullet \beta]} & \quad \text{move down} \\
\frac{[\tau, N \rightarrow \alpha \bullet \beta]}{[\tau, M \rightarrow \bullet \gamma]} & \quad \text{prediction of adjunction} \\
\frac{t \in \text{fSA}(M)}{[\tau, t \rightarrow \bullet R]} & \\
\frac{\tau \in \text{fSA}(N), \text{children}(N) = \gamma}{[\tau, N \rightarrow \alpha M \bullet \beta]} & \quad \text{prediction of adjoined} \\
\frac{[\tau, M \rightarrow \gamma \bullet]}{[\tau, N \rightarrow \alpha M \bullet \beta]} & \quad \text{a possible item (move up)}
\end{align*}
\]
Construction of the automaton (5)

Intuition behind the goto-functions: goto shifts the dot over a node, goto⊥ shifts the dot over a ⊥ (i.e., a foot node daughter).

Definition of goto and goto⊥: Let q be a set of items, M a terminal leaf or a node with $f_{SA}(M) \neq \emptyset$ (no NA constraint),

\[
\begin{align*}
&\triangleright q \in q' \iff \text{goto}(q, M) \neq \emptyset \\
&\triangleright q \in q' \iff \text{goto}_\perp(q, M) \neq \emptyset
\end{align*}
\]

• $\text{goto}(q, M) = \{ [\tau, N \rightarrow \alpha \cdot M \cdot \beta] | [\tau, N \rightarrow \alpha \cdot M \cdot \beta] \in \text{closure}(q) \}$
• $\text{goto}_\perp(q, M) = \{ [\tau, F_t \rightarrow \perp \cdot \beta] | [\tau, F_t \rightarrow \perp \cdot \beta] \in \text{closure}(q) \land t \in f_{SA}(M) \}$

The recognizer (1)

For the definition of the recognizer, we also need the notion of reductions(q) for a given state q.

Intuition: If the closure of q contains a completed item, then the LHS node of the dotted production or, if this is a ⊤ in an auxiliary tree, the whole tree are part of the reductions.

Definition of reductions(q) for a given state q:

\[
\text{reductions}(q) = \{ t \in A | [t, \top \rightarrow R_t \cdot \beta] \in \text{closure}(q) \} \cup
\{ N \in N | [(t, N), N \rightarrow \alpha \cdot \beta] \in \text{closure}(q) \}
\]

Construction of the automaton (6)

Now we can define the set $Q$ of LR states of our automaton as follows:

\[
\begin{align*}
&\triangleright q_\text{fin} \\
&\triangleright q, q' \in Q \iff q' = \text{goto}(q, M) \neq \emptyset \text{ for some node } M \\
&\triangleright q, q' \in Q \iff q' = \text{goto}_\perp(q, M) \neq \emptyset \text{ for some node } M
\end{align*}
\]

A state is final (in $Q_{\text{fin}}$) if its closure contains a completed item for some initial tree:

\[
Q_{\text{fin}} = \{ q \in Q | \text{closure}(q) \cap \{ [t, \top \rightarrow R_t \cdot \beta] | t \in I \} \neq \emptyset \}
\]
The recognizer (3)

Definition of cross-sections $CS(N)$ of a node $N$: Define

$M := N \cup (N \times N^\ast)$

Then for a given node $N$:

- $N \in CS(N)$ if $N$ does not dominate a foot node,
- $(N, L) \in CS(N)$ for each $L \in N^\ast$ if $N$ dominates a foot node,
- $x_1 \ldots x_m \in CS(N)$ if $children(N) = M_1 \ldots M_m$ and $x_i \in CS(M_i)$ for $1 \leq i \leq m$.

Furthermore, $CS^+(N) := CS(N) \setminus ((N) \cup \{(N, L) \mid L \in N^\ast\})$ (the cross-sections without the node itself).

The recognizer (4)

- The stack $\Delta$ contains states and symbols. The latter are either terminal nodes or nonterminal nodes equipped with a stack.
- A configuration $(\Delta, w)$ consists of a stack and a word (the remaining part of the input string).
- There are three operations that allow the automaton to make a transition (i.e., to change configuration): \textit{shift}, \textit{reduce subtree} and \textit{reduce aux tree}.

The recognizer (5)

\textit{shift} pushes the next input symbol followed by a new state on the stack:

$\Delta q \vdash (\Delta q\alpha, aw)$ if $q' = goto(q, a) \neq \emptyset$.

The recognizer (6)

\textit{Reduce subtree} is applied when having completed a subtree rooted in $N$ such that an adjunction occurs at $N$. In other words, it recognizes the part below a foot node.

$(\Delta q_0 X_1 \ldots X_m q_m, w) \vdash (\Delta q_0(\bot, [NL])q', w)$ if

- $N \in reductions(q_m)$, $X_1 \ldots X_m \in CS^+(N)$, $q' = goto_\bot(q_0, N) \neq \emptyset$, and
- $L$ is defined as follows: if some $X_j$ is of the form $(M, L)$, then this provides $L$, otherwise $L = [\ldots]$. 
The recognizer (7)

Reduce aux tree is applied once an auxiliary tree has been recognized. We then go back to the node where the adjunction occurred.

\[(\Delta q_0 X_1 \ldots X_m, w) \vdash (\Delta q'_0 X'_1 \ldots X'_m, w)\] if

- there is a \( t \in reductions(q_m) \) with \( X_1 \ldots X_m \in CS^+(R_t) \),
- \( q' = goto(q_0, N) \neq \emptyset \) where \( N \) is obtained from the unique \( X_j \) of the form \( M[N,L] \), and
- if \( L = [] \), then \( X = N \), otherwise \( X = N[L] \).

The recognizer (8)

- The stack is initialized with the initial state \( q_0 \).
- The stack always contains an alternation of states \( q \in Q \) and nodes or nodes with stacks \( X \in \mathcal{M} \).
- A parse is successful if, in a sequence of transitions (i.e., applications of shift, reduce subtree and reduce aux tree), the input is completely consumed and the automaton reaches a final state:
  Some input \( v \) is recognized if \( (q_m, v) \vdash^* (q_f, \epsilon) \) such that \( q \in Q_f \).