LR(0)-Parser

Attention:
Unfortunately, the LR(0)-parser is in general non-deterministic.

We identify two reasons:

Reduce-Reduce-Conflict:
\[ [A → γ \bullet], \ [A' → γ' \bullet] \in q \hspace{1em} \text{with} \hspace{1em} A \neq A' \lor γ \neq γ' \]

Shift-Reduce-Conflict:
\[ [A → γ \bullet], \ [A' → α \bullet β] \in q \hspace{1em} \text{with} \hspace{1em} α \in T \]
\[ \text{for a state} \hspace{1em} q \in Q. \]

Those states are called LR(0)-unsuited.

Revisiting the Conflicts of the LR(0)-Automaton

What differentiates the particular Reductions and Shifts?

Input:
\[ * 2 + 40 \]

Pushdown:
\[ ( q_0 T ) \]

\[ E \rightarrow E + T \hspace{1em} | \hspace{1em} T \]
\[ T \rightarrow T * F \hspace{1em} | \hspace{1em} F \]
\[ F \rightarrow ( E ) \hspace{1em} | \hspace{1em} \text{int} \]
Revisiting the Conflicts of the LR(0)-Automaton

**Idea:** Matching lookahead with *right context* matters!

**Input:**

\[ 2 + 40 \]

**Pushdown:**

\( \left( q_0 \right) \)

\[
E \rightarrow F + T \\
F \rightarrow T + F | T \\
F^* \rightarrow (E) | \text{int}
\]

\[ \text{name} \]

**LR(k)-Grammars**

**Idea:** Consider \( k \)-lookahead in conflict situations.

**Definition:**

The reduced contextfree grammar \( G \) is called *LR(\( k \))*-grammar, if for \( \text{First}_k(w) = \text{First}_k(x) \) with:

\[
S \rightarrow \alpha A w \rightarrow \alpha \beta w' \quad \text{follows} \quad \alpha = \alpha' \land A = A' \land w' = x
\]

**Strategy for testing Grammars for LR(\( k \))-property**

- Focus iteratively on all rightmost derivations.
- Identify handle \( \alpha \beta \) in sentence forms \( \alpha \beta w \).
- Determine minimal \( k \), such that \( \text{First}_k(w) \) associates \( \beta \) with a unique \( X \rightarrow \beta \) for non-prefixfree \( \alpha \beta S \).
LR(k)-Grammars

for example:

(1) \[ S \rightarrow A \mid B \quad A \rightarrow aAb \mid 0 \quad B \rightarrow aBbb \mid 1 \]

... is not LL(k) for any \( k \):

Let \( S \rightarrow_R \alpha X w \rightarrow \alpha \beta w \). Then \( \alpha \beta \) is of one of these forms:

\[
\begin{array}{c}
\alpha, B, a^n a Ab, a^n aBbb, a^n 0, a^n 1 \\
(a \geq 0)
\end{array}
\]
LR(k)-Grammars

for example:

(1) \[ S \rightarrow A \mid B \quad A \rightarrow a Ab \mid 0 \quad B \rightarrow a Bbb \mid 1 \]

... is not \( LL(k) \) for any \( k \) — but \( LR(0) \):

Let \( S \rightarrow^* \alpha Xw \rightarrow \alpha \beta w \). Then \( \alpha \beta \) is of one of these forms:

\[ A, B, a^n a Ab, a^n a Bbb, a^n 0, a^n 1 \quad (n \geq 0) \]

(2) \[ S \rightarrow a Ac \quad A \rightarrow Abb \mid b \]

LR(k)-Grammars

for example:

(3) \[ S \rightarrow a Ac \quad A \rightarrow b b A \mid b \]

LR(k)-Grammars

for example:

(1) \[ S \rightarrow A \mid B \quad A \rightarrow a Ab \mid 0 \quad B \rightarrow a Bbb \mid 1 \]

... is not \( LL(k) \) for any \( k \) — but \( LR(0) \):

Let \( S \rightarrow^* \alpha Xw \rightarrow \alpha \beta w \). Then \( \alpha \beta \) is of one of these forms:

\[ A, B, a^n a Ab, a^n a Bbb, a^n 0, a^n 1 \quad (n \geq 0) \]

(2) \[ S \rightarrow a Ac \quad A \rightarrow Abb \mid b \]

... is also not \( LL(k) \) for any \( k \):

Let \( S \rightarrow^* \alpha Xw \rightarrow \alpha \beta w \). Then \( \alpha \beta \) is of one of these forms:

\[ ab, a Abb, a Ac \]

LR(k)-Grammars

for example:

(3) \[ S \rightarrow a Ac \quad A \rightarrow b b A \mid b \]

Let \( S \rightarrow^* \alpha Xw \rightarrow \alpha \beta w \) with \( \{y\} = \text{First}_k(w) \) then \( \alpha \beta y \) is of one of these forms:

\[ a b^{2n} b c, a b^{2n} b b A c, a Ac \]
LR(k)-Grammars

for example:

(3)  \[ S \rightarrow a A c \quad A \rightarrow b b A \mid b \quad \text{... is not } LR(0), \text{ but } LR(1): \]
Let \[ S \rightarrow^*_R \alpha X w \rightarrow \alpha \beta w \quad \text{with} \quad \{y\} = \text{First}_k(w) \quad \text{then} \]
\[ \alpha \beta y \quad \text{is of one of these forms:} \]
\[ a b^{2n} b c, \quad a b^{2n} b b A c, \quad a A c \]

LR(k)-Grammars

for example:

(3)  \[ S \rightarrow a A c \quad A \rightarrow b b A \mid b \]
Let \[ S \rightarrow^*_R \alpha X w \rightarrow \alpha \beta w \quad \text{with} \quad \{y\} = \text{First}_k(w) \quad \text{then} \]
\[ \alpha \beta y \quad \text{is of one of these forms:} \]
\[ a b^{2n} b c, \quad a b^{2n} b b A c, \quad a A c \]

LR(k)-Grammars

for example:

(3)  \[ S \rightarrow a A c \quad A \rightarrow b b A \mid b \quad \text{... is not } LR(0), \text{ but } LR(1): \]
Let \[ S \rightarrow^*_R \alpha X w \rightarrow \alpha \beta w \quad \text{with} \quad \{y\} = \text{First}_k(w) \quad \text{then} \]
\[ \alpha \beta y \quad \text{is of one of these forms:} \]
\[ a b^{2n} b c, \quad a b^{2n} b b A c, \quad a A c \]

(4)  \[ S \rightarrow a A c \quad A \rightarrow b A b \mid b \]

Consider the rightmost derivations:
\[ S \rightarrow^*_R a b^n A b^n c \rightarrow a b^n b b^n c \]
LR(k)-Grammars

for example:

(3) \[ S \rightarrow a A c \quad A \rightarrow b b A \mid b \] is not LR(0), but LR(1):

Let \[ S \rightarrow_{\gamma} \alpha X w \rightarrow \alpha \beta w \] with \( \{y\} = \text{First}_{\kappa}(w) \) then \( \alpha \beta y \) is of one of these forms:

\[ a b^{2n} b c, a b^{2n} b b A c, a A c \]

(4) \[ S \rightarrow a A c \quad A \rightarrow b A b \mid b \] is not LR(k) for any \( k \geq 0 \):

Consider the rightmost derivations:

\[ S \rightarrow_{R} a b^n A b^n c \rightarrow a b^n b b^n e \]

LR(1)-Parsing

Idea: Let’s equip items with 1-lookahead

Admissible LR(1)-Items

The item \([B \rightarrow \alpha \bullet \beta, x]\) is admissible for \( \gamma \alpha \) if:

\[ S \rightarrow_{R} \gamma B w \quad \text{with} \quad \{x\} = \text{First}_{1}(w) \]
The Characteristic LR(1)-Automaton

The set of admissible LR(1)-items for viable prefixes is again computed with the help of the finite automaton $c(G, 1)$.

The automaton $c(G, 1)$:

**States:** LR(1)-items

**Start state:** $[S' \to \bullet S, \epsilon]$

**Final states:** $\{[B \rightarrow \gamma \bullet, x] \mid B \rightarrow \gamma \in P, x \in \text{Follow}_1(B)\}$

**Transitions:**

1. $((A \rightarrow \alpha \bullet X \beta, x), X, [A \rightarrow \alpha X \bullet \beta, x]), \ X \in (N \cup T)$
2. $((A \rightarrow \alpha \bullet B \beta x), B, [B \rightarrow \bullet \gamma, x'], A \rightarrow \alpha \beta B, B \rightarrow \gamma \in P, x' \in \text{First}_1(\beta) \circ \{x\}$

The Canonical LR(1)-Automaton

The canonical LR(1)-automaton $LR(G, 1)$ is created from $c(G, 1)$, by performing arbitrarily many $\epsilon$-transitions and then making the resulting automaton deterministic ...

The Canonical LR(1)-Automaton

The canonical LR(1)-automaton $LR(G, 1)$ is created from $c(G, 1)$, by performing arbitrarily many $\epsilon$-transitions and then making the resulting automaton deterministic ...

But again, it can be constructed directly from the grammar; analogously to LR(0), we need the $\epsilon$-closure $\delta_\epsilon^*$ as a helper function:

$$\delta_\epsilon^*(g) = \emptyset \cup \{C \rightarrow \bullet \epsilon, x' \mid \exists [A \rightarrow \alpha \bullet B \beta', x'] \in g, \beta' \in (N \cup T)^* : B \rightarrow \emptyset, A \rightarrow \alpha B \beta', x' \in \text{First}_1(\beta \beta') \circ \{x\}\}$$

The Characteristic LR(1)-Automaton

The set of admissible LR(1)-items for viable prefixes is again computed with the help of the finite automaton $c(G, 1)$.

The automaton $c(G, 1)$:

**States:** LR(1)-items

**Start state:** $[S' \to \bullet S, \epsilon]$

**Final states:** $\{[B \rightarrow \gamma \bullet, x] \mid B \rightarrow \gamma \in P, x \in \text{Follow}_1(B)\}$

**Transitions:**

1. $((A \rightarrow \alpha \bullet X \beta, x), X, [A \rightarrow \alpha X \bullet \beta, x]), \ X \in (N \cup T)$
2. $((A \rightarrow \alpha \bullet B \beta x), B, [B \rightarrow \bullet \gamma, x'], A \rightarrow \alpha \beta B, B \rightarrow \gamma \in P, x' \in \text{First}_1(\beta) \circ \{x\}$

This automaton works like $c(G)$ — but additionally manages a 1-prefix from $\text{Follow}_1$ of the left-hand sides.
The Canonical LR(1)-Automaton

The canonical LR(1)-automaton $LR(G, 1)$ is created from $G(G, 1)$, by performing arbitrarily many $\epsilon$-transitions and then making the resulting automaton deterministic ...

But again, it can be constructed directly from the grammar; analogously to $LR(0)$, we need the $\epsilon$-closure $\delta^*_{\epsilon}$ as a helper function:

$$\delta^*_{\epsilon}(q) = q \cup \{ [C \rightarrow \gamma, x] \mid \exists [A \rightarrow \alpha \cdot B \beta', x'] \in q, \beta \in (N \cup T)^* : \beta \rightarrow^* C \beta' \land x \in \text{First}_1(\beta \beta') \otimes \{ x' \} \}$$

Then, we define:

- States: Sets of LR(1)-items;
- Start state: $\delta^*_{\epsilon} \{ [S' \rightarrow \cdot S, \epsilon] \}$
- Final states: $\{ q \mid \exists [A \rightarrow \alpha \in P : \ [A \rightarrow \alpha \cdot, x] \in q \}$
- Transitions: $\delta(q, X) = \delta^*_{\epsilon} \{ [A \rightarrow \alpha \cdot X \cdot \beta, x] \mid [A \rightarrow \alpha \cdot, x] \in q \}$

For example:

$$
\begin{align*}
E & \rightarrow E + T \mid T \\
T & \rightarrow T \ast F \mid F \\
F & \rightarrow (E) \mid \text{int}
\end{align*}
$$

$\text{First}_1(S') = \text{First}_1(E) = \text{First}_1(T) = \text{First}_1(F) = \text{name, int, (}$

$$
\begin{align*}
q_0 &= \{ [S' \rightarrow \cdot E, \{ \{ \}, \} ], [E \rightarrow \cdot E + T, \{ \{ \}, +\} ], [T \rightarrow \cdot T \ast F, \{ \{ \}, \ast\} ], [F \rightarrow \cdot (E), \{ \{ \}, \{ \} \} ] \} \\
q_3 &= \delta(q_0, F) = \{ [T \rightarrow \cdot F \ast ], [F \rightarrow \cdot F \ast ] \} \\
q_4 &= \delta(q_0, \text{int}) = \{ [F \rightarrow \cdot \text{int} \ast ], [\text{int} \rightarrow \cdot \text{int} \ast ] \} \\
q_5 &= \delta(q_0, () ) = \{ [F \rightarrow \cdot (\{ \}), [\text{int} \rightarrow \cdot (\{ \}) ] \} \\
q_1 &= \delta(q_0, E) = \{ [S' \rightarrow \cdot E, \{ \{ \}, \} ], [E \rightarrow \cdot E + T, \{ \{ \}, +\} ], [T \rightarrow \cdot T \ast F, \{ \{ \}, \ast\} ], [F \rightarrow \cdot (E), \{ \{ \}, \{ \} \} ] \} \\
q_2 &= \delta(q_0, T) = \{ [E \rightarrow \cdot T, [T \rightarrow \cdot T \ast F, \{ \{ \}, \ast\} ] ] \}
\end{align*}
$$
First_1(S') = First_1(E) = First_1(T) = First_1(F) = name, int, (  

q_0 = δ(q_0, E) = \{[S' \rightarrow E \cdot \{\varepsilon\}], [E \rightarrow E \cdot T, \{\varepsilon, +\}], [E \rightarrow \varepsilon], [T \rightarrow T \cdot F, \{\varepsilon, +, *\}], [T \rightarrow \varepsilon], [F \rightarrow F \cdot \{\varepsilon, +, *\}], [F \rightarrow F\cdot T, \{\varepsilon, +\}]\}  

q_3 = δ(q_0, E) = \{[T \rightarrow F \cdot \{\varepsilon, +, *\}], [T \rightarrow \varepsilon], [F \rightarrow F \cdot \{\varepsilon, +, *\}], [F \rightarrow F\cdot T, \{\varepsilon, +\}]\}  

q_4 = δ(q_0, int) = \{[F \rightarrow \varepsilon], [F \rightarrow F \cdot \{\varepsilon, +, *\}], [F \rightarrow F\cdot T, \{\varepsilon, +\}]\}  

q_5 = δ(q_0, ( ) = \{[F \rightarrow F \cdot \{\varepsilon, +, *\}], [F \rightarrow F\cdot T, \{\varepsilon, +\}]\}  

\text{The Canonical LR(1)-Automaton}
The Canonical LR(1)-Automaton

Discussion:
- In the example, the number of states was almost doubled ... and it can become even worse
- The conflicts in states $q_1, q_2, q_0$ are now resolved! e.g. we have for:

$$q_1 = \left\langle \left\{ [E \rightarrow E + T], \{\epsilon, +, \cdot\} \right\}, \left\{ [T \rightarrow T * F], \{\epsilon, +, \cdot\} \right\} \right\rangle$$

with:

$$\{\epsilon, +\} \cap \text{First}(\{F\}) \cap \{\epsilon, +, \cdot\} = \{\epsilon, +\} \cap \{\cdot\} = \emptyset$$

The LR(1)-Parser:

Possible actions are:
- **shift**
- **reduce** $(A \rightarrow \gamma)$
- **error**

Shift-operation
Reduction with callback/output
Error

... for example:

<table>
<thead>
<tr>
<th>action</th>
<th>$\epsilon$</th>
<th>int</th>
<th>( )</th>
<th>$+$</th>
<th>$\ast$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$S',0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td>$E,1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>$T,1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_4$</td>
<td>$T,1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_5$</td>
<td>$F,1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_6$</td>
<td>$F,1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_7$</td>
<td>$E,0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_8$</td>
<td>$E,0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_9$</td>
<td>$T,0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_{10}$</td>
<td>$T,0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_{11}$</td>
<td>$F,0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_{12}$</td>
<td>$F,0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The goto-table encodes the transitions:
$$\text{goto}[q, X] = \delta(q, X) \in Q$$

The action-table describes for every state $q$ and possible lookahead $w$ the necessary action.
The LR(1)-Parser

The construction of the \( LR(1) \)-parser:

States: \( Q \cup \{ f \} \) \hspace{1cm} \( (f \text{ fresh}) \)

Start state: \( q_0 \)

Final state: \( f \)

Transitions:

Shift: \( (p, a, p q) \) if \( q = \text{goto}(q, a), \) \( s = \text{action}(p, w) \)

Reduce: \( (p q_1 \ldots q_{|\beta|}, \epsilon, p q) \) if \( [A \rightarrow \beta \bullet] \in q_{|\beta|}, \) \( q = \text{goto}(p, A), \) \( [A \rightarrow \beta \bullet = \text{action}]_{|q_{|\beta|}, w} \)

Finish: \( (q_0, p, \epsilon, f) \) if \( [S' \rightarrow S \bullet] \in p \)

with \( LR(G, 1) = (Q, T, \delta, q_0, F) \).

The Canonical LR(1)-Automaton

In general:

We identify two conflicts:

Reduce-Reduce-Conflict:
\[
[A \rightarrow \gamma \bullet, x], \ [A' \rightarrow \gamma' \bullet, x] \in q \quad \text{with} \quad A \neq A' \cup \gamma \neq \gamma'
\]

Shift-Reduce-Conflict:
\[
[A \rightarrow \gamma \bullet, x], \ [A' \rightarrow \alpha \bullet a \beta, y] \in q \quad \text{with} \quad a \in T \cup x \in \{a\} \quad \text{for a state} \quad q \in Q.
\]

Such states are now called \( LR(1) \)-unsuited

Special LR(k)-Subclasses

Theorem:

A reduced context-free grammar \( G \) is called \( LR(k) \) iff the canonical \( LR(k) \)-automaton \( LR(G, k) \) has no \( LR(k) \)-unsuited states.

A reduced context-free grammar \( G \) is called \( LR(k) \) iff the canonical \( LR(k) \)-automaton \( LR(G, k) \) has no \( LR(k) \)-unsuited states.
Special LR(k)-Subclasses

**Theorem:**
A reduced contextfree grammar $G$ is called $LR(k)$ iff the canonical $LR(k)$-automaton $LR(G,k)$ has no $LR(k)$-unsuited states.

**Discussion:**
- Our example apparently is $LR(1)$
- In general, the canonical $LR(k)$-automaton has much more states then $LR(G) = LR(G,0)$
- Therefore in practice, subclasses of $LR(k)$-grammars are often considered, which only use $LR(G)$ ...

**Parsing Methods**

- deterministic languages
  - $LR(1) = \ldots = LR(k)$
  - LALR(k)
  - SLR(k)
  - LR(0)
  - [Regular languages]
    - LL(1)
    - LL(k)

Lexical and Syntactical Analysis:

- Concept of specification and implementation:

  $E \rightarrow E op E$

  Generator

  $E \rightarrow [1-9][0-9]^*$

  Generator

  $0, 1-9, [0-9]$
Lexical and Syntactical Analysis:

From Regular Expressions to Finite Automata

From Finite Automata to Scanners

Lexical and Syntactical Analysis:

Computation of lookahead sets:

From Item-Pushdown Automata to LL(1)-Parsers:

Lexical and Syntactical Analysis:

From characteristic to canonical Automata:

From Shift-Reduce-Parsers to LR(1)-Parsers:

Topic:
Semantic Analysis