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 a helper function:

$$\delta_5^*(q) = q \cup \{ [B \rightarrow \gamma, \epsilon] \mid \exists [A \rightarrow \alpha \bullet B' \beta', \epsilon] \in q, \beta \in (N \cup T)^* :$$

$$B' \rightarrow^* B \beta \wedge x \in \text{First}_1(\beta \beta') \cup \{ \epsilon' \} \}$$

Then, we define:

- **States:** Sets of $LR(1)$-items;
- **Start state:** $\delta_5^*([S' \rightarrow S, \epsilon])$;
- **Final states:** $\{ q \mid \exists A \rightarrow \alpha \in P : [A \rightarrow \alpha \bullet, \epsilon] \in q \}$
- **Transitions:** $\delta(q, X) = \delta_5^* \{ [A \rightarrow \alpha X \bullet \beta, \epsilon] \mid [A \rightarrow \alpha \bullet X \beta, \epsilon] \in q \}$

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 a helper function:

$$\delta_5^*(q) = q \cup \{ [B \rightarrow \gamma, \epsilon] \mid \exists [A \rightarrow \alpha \bullet B' \beta', \epsilon] \in q, \beta \in (N \cup T)^* :$$

$$B' \rightarrow^* B \beta \wedge x \in \text{First}_1(\beta \beta') \cup \{ \epsilon' \} \}$$

Then, we define:

- **States:** Sets of $LR(1)$-items;
- **Start state:** $\delta_5^*([S' \rightarrow S, \epsilon])$;
- **Final states:** $\{ q \mid \exists A \rightarrow \alpha \in P : [A \rightarrow \alpha \bullet, \epsilon] \in q \}$
- **Transitions:** $\delta(q, X) = \delta_5^* \{ [A \rightarrow \alpha X \bullet \beta, \epsilon] \mid [A \rightarrow \alpha \bullet X \beta, \epsilon] \in q \}$
The canonical LR(1)-automaton

The canonical LR(1)-automaton $LR(G, 1)$ is created from $c(G, 1)$, by performing arbitrarily many $\varepsilon$-transitions and then making the resulting automaton deterministic. But again, it can be constructed directly from the grammar.

Analogously to $LR(0)$, we need a helper function:

$$\delta_{\varepsilon}^{\star}(q) = q \cup \{ [B \rightarrow \bullet \gamma, x] \mid \exists [A \rightarrow \alpha \bullet B' \beta', x'] \in q, \beta \in (N \cup T)^* : B' \rightarrow^* B \beta \land x \in \text{First}_{\varepsilon}(\beta \beta') \cap \{x'\} \}$$

Then, we define:

- **States**: Sets of $LR(1)$-items;
- **Start state**: $\delta_{\varepsilon}^{\star}\{[S \rightarrow S, \varepsilon]\}$
- **Final states**: $\{q \mid \exists A \rightarrow \alpha \in P : [A \rightarrow \alpha \bullet x] \in q\}$
- **Transitions**: $\delta(q, X) = \delta_{\varepsilon}^{\star}\{[A \rightarrow \alpha X \bullet \beta, x] \mid [A \rightarrow \alpha X \beta, x] \in q\}$
The canonical LR(1)-automaton

The canonical LR(G,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 LRT(0), we need a helper function:

$$\delta_\epsilon^*: q \cup \{ [B \rightarrow \alpha \bullet B', x'] \mid \exists \beta \in (N \cup T)^*: B' \rightarrow B \beta \land x \in \text{First}_i(B \beta') \cup \{ x' \} \}$$

Then, we define:

**States:** Sets of LR(1)-items;
**Start state:** $\delta^*_0 \{ [S' \rightarrow S, \epsilon] \}$
**Final states:** $q \cup \{ [A \rightarrow \alpha \in P : [A \rightarrow \alpha \bullet, x] \in q} \}$
**Transitions:** $\delta(q, X) = \delta^*_\epsilon \{ [A \rightarrow \alpha X \bullet \beta, x] \mid [A \rightarrow \alpha \bullet X \beta, x] \in q \}$

The canonical LR(1)-automaton

The canonical LR(G,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 a helper function:

$$\delta_\epsilon^*(q) = q \cup \{ [B \rightarrow \alpha \gamma, x] \mid \exists [A \rightarrow \alpha \bullet B', x'] \in q, \beta \in (N \cup T)^*: B' \rightarrow B \beta \land x \in \text{First}_i(B \beta') \cup \{ x' \} \}$$

Then, we define:

**States:** Sets of LR(1)-items;
**Start state:** $\delta^*_0 \{ [S' \rightarrow S, \epsilon] \}$
**Final states:** $q \cup \{ [A \rightarrow \alpha \in P : [A \rightarrow \alpha \bullet, x] \in q} \}$
**Transitions:** $\delta(q, X) = \delta^*_\epsilon \{ [A \rightarrow \alpha X \bullet \beta, 0] \mid [A \rightarrow \alpha \bullet X \beta, 0] \in q \}$
The canonical LR(1)-Automaton

for example:

\[
q_5' = \delta(q_5, \epsilon) = \delta(q_5, \epsilon) = \emptyset \]

\[
q_6 = \delta(q_6, +) = \emptyset \]

Discussion:

- In the example, the number of states was almost doubled ... and it can become even worse

- The conflicts in states \(q_1, q_2, q_3, q_6\) are now resolved!
  e.g. we have for:

\[
q_0 = \begin{cases}
\{E \rightarrow E + T \}, \\
\{T \rightarrow T \ast F\} \\
\{F \rightarrow \epsilon + \ast\}
\end{cases}
\]

with:

\[
\{\epsilon, \ast\} \cap \text{First}_1(\ast F) \cap \{\epsilon, \ast, \ast\} = \emptyset
\]
The LR(1)-Parser:

**The LR(1)-Parser:**

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

Shift-operation
- Reduction with callback/output
- Error

... for example:

\[
E \rightarrow E + T^0 \mid T^1 \\
T \rightarrow T * F^0 \mid F^1 \\
F \rightarrow (E)^0 \mid \text{int}^1
\]

The action-table describes for every state $q$ and possible lookahead $w$ the necessary action.

The Canonical LR(1)-Automat

In general:

We identify two conflicts:

Reduce-Reduce-Conflict:

$[A \rightarrow \gamma \cdot x], \ [A' \rightarrow \gamma' \cdot \cdot x] \in q$ with $A \neq A' \lor \gamma \neq \gamma'$

Shift-Reduce-Conflict:

$[A \rightarrow \gamma \cdot x], \ [A' \rightarrow \alpha \cdot a \beta \cdot x] \in q$

with $a \in T$ und $x \in \{a\}$.

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

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 Canonical LR(1)-Automat

In general:

We identify two conflicts:

Reduce-Reduce-Conflict:

\[ [A \rightarrow \gamma \bullet, x], \quad [A' \rightarrow \gamma' \bullet, x] \in q \quad \text{with} \quad A \neq A' \lor \gamma \neq \gamma' \]

Shift-Reduce-Conflict:

\[ [A \rightarrow \gamma \bullet, x], \quad [A' \rightarrow A \beta \cdot y, y] \in q \quad \text{with} \quad a \in T \quad \text{and} \quad x \in \{a\} \cup \text{First}(\beta) \cap \{y\} \]

for a state \( q \in Q \).

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

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 \( LR(G) \) ...

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 \( LR(G) \) ...
- For resolving conflicts, the items are assigned special lookahead-sets:
  - independently on the state itself \( \Rightarrow \text{Simple LR(k)} \)
  - dependent on the state itself \( \Rightarrow \text{LALR(k)} \)
Chapter 5:
Summary

Lexical and Syntactical Analysis:

From Regular Expressions to Finite Automata

From Finite Automata to Scanners

Parsing Methods

- Deterministic languages: L(1) ⊆ L(2) ⊆ ..., L(k)
- LR(k) grammars describe all prefix-free deterministic context-free languages
- The language-classes of LL(k)-grammars form a hierarchy within the deterministic context-free languages.

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: