Characteristic Automaton

Observation:
The set of viable prefixes from \((N \cup T)^*\) for (admissible) items can be computed from the content of the shift-reduce parser’s pushdown with the help of a finite automaton:

**States:** Items

**Start state:** \([S \rightarrow \bullet S]\)

**Final states:** \([B \rightarrow \gamma \bullet \mid B \rightarrow \gamma \in P]\)

**Transitions:**

1. \( (\alpha \rightarrow \alpha \bullet X \beta, X, [A \rightarrow \alpha X \bullet \beta]), \quad X \in (N \cup T), A \rightarrow \alpha X \beta \in P; \)
2. \( (\alpha \rightarrow \alpha \bullet B \beta, \epsilon, [B \rightarrow \bullet \gamma]), \quad A \rightarrow \alpha B \beta, \quad B \rightarrow \gamma \in P; \)

The automaton \(c(G)\) is called characteristic automaton for \(G\).
Canonical LR(0)-Automaton

The canonical LR(0)-automaton $LR(G)$ is created from $G(G)$ by:
- performing arbitrarily many $ε$-transitions after every consuming transition
- performing the powerset construction

... for example:

Canonical LR(0)-Automaton

The canonical LR(0)-automaton $LR(G)$ is created from $G(G)$ by:
- performing arbitrarily many $ε$-transitions after every consuming transition
- performing the powerset construction

... for example:

Canonical LR(0)-Automaton

Therefore we determine:

$q_0 = \delta(q_0, E) = \{ [S' \to E], [E \to E + T], [E \to T], [T \to T * F] \}$
$q_1 = \delta(q_0, E) = \{ [S' \to E], [E \to E + T] \}$
$q_2 = \delta(q_0, T) = \{ [T \to T * F] \}$
$q_3 = \delta(q_0, F) = \{ [F \to F *] \}$
$q_4 = \delta(q_0, int) = \{ [F \to int *] \}$

Canonical LR(0)-Automaton

$q_3 = \delta(q_0, ( ) ) = \{ [F \to ( E ) ] \}$
$q_7 = \delta(q_2, * ) = \{ [F \to E *], [T \to T * F] \}$
$q_8 = \delta(q_5, E) = \{ [F \to ( E * ) ] \}$
$q_9 = \delta(q_2, E) = \{ [T \to E *], [E \to E + T] \}$
$q_{10} = \delta(q_7, T) = \{ [T \to T * F] \}$
$q_{11} = \delta(q_6, ) ) = \{ [F \to E *] \}$
Canonical LR(0)-Automaton

The canonical LR(0)-automaton LR(G) is created from ε(G) by:
- performing arbitrarily many ε-transitions after every consuming transition
- performing the powerset construction

... for example:

Canonical LR(0)-Automaton

Therefore we determine:

q₀ = \{[S' \rightarrow E], [E \rightarrow \bullet E + T], [E \rightarrow \bullet T], [T \rightarrow \bullet T * F], [T \rightarrow \bullet F], [F \rightarrow \bullet (E)], [F \rightarrow \bullet \text{int}!]\}
q₁ = δ(q₀, E) = \{[S' \rightarrow E \bullet], [E \rightarrow E \bullet + T], [E \rightarrow \bullet T], [T \rightarrow T \bullet * F], [T \rightarrow \bullet F], [F \rightarrow \bullet (E)], [F \rightarrow \bullet \text{int}!]\}
q₂ = δ(q₀, T) = \{[E \rightarrow E + T \bullet], [E \rightarrow \bullet T \bullet], [T \rightarrow T \bullet * F], [T \rightarrow \bullet F], [F \rightarrow \bullet (E)], [F \rightarrow \bullet \text{int}!]\}
q₃ = δ(q₀, F) = \{[T \rightarrow T \bullet], [T \rightarrow \bullet F], [F \rightarrow \bullet (E)], [F \rightarrow \bullet \text{int}!]\}
q₄ = δ(q₀, \text{int}) = \{[F \rightarrow \text{int}!]\}

Canonical LR(0)-Automaton

... for example:

Canonical LR(0)-Automaton

Therefore we determine:

q₀ = \{[S' \rightarrow E], [E \rightarrow \bullet E + T], [E \rightarrow \bullet T], [T \rightarrow \bullet T * F], [T \rightarrow \bullet F], [F \rightarrow \bullet (E)], [F \rightarrow \bullet \text{int}!]\}
q₁ = δ(q₀, E) = \{[S' \rightarrow E \bullet], [E \rightarrow E \bullet + T], [E \rightarrow \bullet T], [T \rightarrow T \bullet * F], [T \rightarrow \bullet F], [F \rightarrow \bullet (E)], [F \rightarrow \bullet \text{int}!]\}
q₂ = δ(q₀, T) = \{[E \rightarrow E + T \bullet], [E \rightarrow \bullet T \bullet], [T \rightarrow T \bullet * F], [T \rightarrow \bullet F], [F \rightarrow \bullet (E)], [F \rightarrow \bullet \text{int}!]\}
q₃ = δ(q₀, F) = \{[T \rightarrow T \bullet], [T \rightarrow \bullet F], [F \rightarrow \bullet (E)], [F \rightarrow \bullet \text{int}!]\}
q₄ = δ(q₀, \text{int}) = \{[F \rightarrow \text{int}!]\}
LR(0)-Parser

Idea for a parser:

- The parser manages a viable prefix \( \alpha = X_1 \ldots X_m \) on the pushdown and uses \( LR(G) \), to identify reduction spots.
- It can reduce with \( A \rightarrow \gamma \), if \( [A \rightarrow \gamma \bullet] \) is admissible for \( \alpha \).

Optimization:

We push the states instead of the \( X_i \) in order not to process the pushdown's content with the automaton anew all the time.
Reduction with \( A \rightarrow \gamma \) leads to popping the uppermost \( |\gamma| \) states and continue with the state on top of the stack and input \( A \).

Attention:

This parser is only deterministic, if each final state of the canonical \( LR(0) \)-automaton is conflict free.

---

 Canonical LR(0)-Automaton

Observation:

The canonical \( LR(0) \)-automaton can be created directly from the grammar.
Therefore we need a helper function \( \delta^* \) (\( \epsilon \)-closure)

We define:

**States:** Sets of items;

- Start state: \( \{ [S' \rightarrow \bullet S] \} \)
- Final state: \( q \) \( \mid \exists A \rightarrow \alpha \in P : \ [A \rightarrow \alpha \bullet] \in q \)
- Transitions: \( \delta^* \) \( \{ [A \rightarrow \alpha X \bullet \beta] \mid [A \rightarrow \alpha \bullet X \beta] \in q \} \)

---

LR(0)-Parser

... for example:

\[
q_1 = \{ [S' \rightarrow E \bullet], \quad E \rightarrow E \bullet + T \} \]
\[
q_2 = \{ [E \rightarrow T \bullet], \quad T \rightarrow T \bullet + F \} \quad q_3 = \{ [T \rightarrow F \bullet] \}
\]
\[
q_4 = \{ [T \rightarrow F \bullet] \}
\]

The final states \( q_5, q_3, q_4 \) contain more then one admissible item

\[ \Rightarrow \text{non deterministic!} \]

---

The construction of the \( LR(0) \)-parser:

**States:** \( Q \cup \{ f \} \) \( (f \) fresh)

**Start state:** \( q_0 \)

**Final state:** \( f \)

**Transitions:**

**Shift:** \( (p, p, p q) \) if \( q = \delta(p, a) \neq \emptyset \)

**Reduce:** \( (p q_1 \ldots q_m, \epsilon, p q) \) if \( [A \rightarrow X_1 \ldots X_m \bullet] \in q_m, \)

\[ q = \delta(p, A) \]

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

with \( LR(G) = (Q, T, \delta, q_0, F) \).
LR(0)-Parser

Correctness:

we show:

The accepting computations of an LR(0)-parser are one-to-one related to those of a shift-reduce parser $M^R_G$.

we conclude:

- The accepted language is exactly $L(G)$
- The sequence of reductions of an accepting computation for a word $w \in T$ yields a reverse rightmost derivation of $G$ for $w$

LR(0)-Parser

Attention:

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

We identify two reasons:

- **Reduce-Reduce-Conflict:**
  $$[A \rightarrow \gamma \bullet] \cdot [A' \rightarrow \gamma' \bullet] \in q \quad \text{with} \quad A \neq A' \lor \gamma \neq \gamma'$$

- **Shift-Reduce-Conflict:**
  $$[A \rightarrow \gamma \bullet] \cdot [A' \rightarrow \alpha \bullet \beta] \in q \quad \text{with} \quad \alpha \in T$$

for a state $q \in Q$.

Those states are called LR(0)-unsuit.
LR(k)-Grammar

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 \( k \) — but LR(0):
   Let \( S \rightarrow_\ast \alpha Xw \rightarrow \alpha \beta w \). Then \( \alpha \beta \) is of one of these forms:
   \[ A, B, a^n aAb, a^n aBBb, a^n 0, a^n 1 \ (n \geq 0) \]

2. \( S \rightarrow aAc \quad A \rightarrow Abb \mid b \)
   ... is also LR(0):
   Let \( S \rightarrow_\ast \alpha Xw \rightarrow \alpha \beta w \). Then \( \alpha \beta \) is of one of these forms:
   \[ ab, aAbb, aAc \]

3. \( S \rightarrow aAc \quad A \rightarrow bbA \mid b \)
   ... is not LR(0), but LR(1):
   Let \( S \rightarrow_\ast \alpha Xw \rightarrow \alpha \beta w \) with \( \{y\} = \text{First}_k(w) \) then
   \( \alpha \beta y \) is of one of these forms:
   \[ ab^{2n}bc, ab^{2n}bbAc, aAc \]

4. \( S \rightarrow aAc \quad A \rightarrow bAb \mid b \)
   ... is not LR(k) for any \( k \geq 0 \):
   Consider the rightmost derivations:
   \[ S \rightarrow_\ast \alpha b^n Ab^n c \rightarrow_\ast \alpha \]

LR(k)-Grammar

for example:

3. \( S \rightarrow aAc \quad A \rightarrow bbA \mid b \)
   ... is not LR(0), but LR(1):
   Let \( S \rightarrow_\ast \alpha Xw \rightarrow \alpha \beta w \) with \( \{y\} = \text{First}_k(w) \) then
   \( \alpha \beta y \) is of one of these forms:
   \[ ab^{2n}bc, ab^{2n}bbAc, aAc \]

4. \( S \rightarrow aAc \quad A \rightarrow bAb \mid b \)
   ... is not LR(k) for any \( k \geq 0 \):
   Consider the rightmost derivations:
   \[ S \rightarrow_\ast \alpha b^n Ab^n c \rightarrow_\ast \alpha \]

The Characteristic LR(1)-Automaton

The automaton \( c(G, 1) \):

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

**Start state:** \([S' \rightarrow S, \epsilon]\)

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

**Transitions:**

1. \((A \rightarrow \alpha \bullet X \beta, x, X, [A \rightarrow \alpha X \bullet \beta, x]), \quad X \in (N \cup T)\)

2. \((A \rightarrow \alpha \bullet B \beta, x, \epsilon, [B \rightarrow \bullet \gamma, x']), \quad A \rightarrow \alpha B \beta, \quad B \rightarrow \gamma \in P, x' \in \text{First}(\beta) \cup \{x\}\)

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

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 ...

\[ \delta^*_\epsilon(q) = q \cup [B \rightarrow \bullet \gamma, x]\]

\[ \exists [A \rightarrow \alpha \bullet B' \beta', x'] \in q, \beta \in (N \cup T)^* : B' \rightarrow^* B \beta \land x \in \text{First}(\beta \beta') \cap \{x'\}\]
The canonical LR(1)-automaton

The canonical LR(1)-automaton \( LR(G, 1) \) is created from \( e(G, 1) \), by performing arbitrarily many \( e \)-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:

\[
\delta^*_{e}(q) = q \cup \{ [B \rightarrow \bullet \gamma, x] \mid \exists [A \rightarrow \alpha \bullet B' \beta', x'] \in q, B \in (N \cup T)^* : B' \rightarrow^* B \beta \land x \in \text{First}_e(\beta \beta') \ominus \{ x' \} \}
\]

Then, we define:

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