2.6 Bottom-up Analyse

Achtung:

- Viele Grammatiken sind nicht $LL(k)$:-(
- Eine Grund ist Links-Rekursivität ...
- Die Grammatik G heißt links-rekursiv, falls

$$A \rightarrow^{+} A\,\beta \qquad \text{für ein} \quad A \in N,\ \beta \in (T \cup N)^{*}$$

2.6 Bottom-up Analyse

Achtung:

- Viele Grammatiken sind nicht $LL(k)$:-(
- Eine Grund ist Links-Rekursivität ...
- Die Grammatik G heißt links-rekursiv, falls

$$A \rightarrow^{+} A\,\beta \qquad \text{für ein} \quad A \in N,\ \beta \in (T \cup N)^{*}$$

Beispiel:

$$\begin{array}{lcl} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & (\ E\ ) \mid \mathsf{name} \mid \mathsf{int} \end{array}$$

... ist links-rekursiv :-)

Satz

Ist die Grammatik G reduziert und links-rekursiv, dann ist G nicht $LL(k)$ für jedes k.

Satz

Ist die Grammatik G reduziert und links-rekursiv, dann ist G nicht $LL(k)$ für jedes k.

Beweis: Vereinfachung: $A \to A\,\beta \in P$

A erreichbar $\Longrightarrow$ $S \to_L^* u\,A\,\gamma \to_L^* u\,A\,\beta^n\,\gamma$ für jedes $n \geq 0$.

A produktiv $\Longrightarrow$ $\exists\, A \to \alpha : \alpha \neq A\,\beta$.

Satz

Ist die Grammatik G reduziert und links-rekursiv, dann ist G nicht $LL(k)$ für jedes k.

Beweis: Vereinfachung: $A \to A\,\beta \in P$

A erreichbar $\Longrightarrow$ $S \to_L^* u\,A\,\gamma \to_L^* u\,A\,\beta^n\,\gamma$ für jedes $n \geq 0$.

A produktiv $\Longrightarrow$ $\exists\, A \to \alpha : \alpha \neq A\,\beta$.

Annahme: G ist $LL(k)$;-) Dann gilt für alle $n \geq 0$:

$$\mathsf{First}_k(\alpha\,\beta^n\,\gamma) \cap \mathsf{First}_k(A\,\beta\,\beta^n\,\gamma) = \emptyset$$

Weil $\mathsf{First}_k(\alpha\,\beta^{n+1}\,\gamma) \subseteq \mathsf{First}_k(A\,\beta^{n+1}\,\gamma)$

folgt: $\mathsf{First}_k(\alpha\,\beta^n\,\gamma) \cap \mathsf{First}_k(\alpha\,\beta^{n+1}\,\gamma) = \emptyset$

Satz

Ist die Grammatik G reduziert und links-rekursiv, dann ist G nicht $LL(k)$ für jedes k.

Beweis: Vereinfachung: $A \to A\,\beta \in P$

A erreichbar $\Longrightarrow$ $S \to_L^* u\,A\,\gamma \to_L^* u\,A\,\beta^n\,\gamma$ für jedes $n \geq 0$.

A produktiv $\Longrightarrow$ $\exists\, A \to \alpha : \alpha \neq A\,\beta$.

Annahme: G ist $LL(k)$;-) Dann gilt für alle $n \geq 0$:

$$\begin{array}{lcccl} & \mathsf{First}_k(\alpha\,\beta^n\,\gamma) & \cap & \mathsf{First}_k(A\,\beta\,\beta^n\,\gamma) & = \emptyset \\ \text{Weil} & \mathsf{First}_k(\alpha\,\beta^{n+1}\,\gamma) & \subseteq & \mathsf{First}_k(A\,\beta^{n+1}\,\gamma) & \\ \text{folgt:} & \mathsf{First}_k(\alpha\,\beta^n\,\gamma) & \cap & \mathsf{First}_k(\alpha\,\beta^{n+1}\,\gamma) & = \emptyset \end{array}$$

Fall 1: $\beta \to^* \epsilon$ — Widerspruch !!!

Fall 2: $\beta \to^* w \neq \epsilon \Longrightarrow \mathsf{First}_k(\alpha\,\beta^k\,\gamma) \cap \mathsf{First}_k(\alpha\,\beta^{k+1}\,\gamma) \neq \emptyset$:-(

Bottom-up Parsing:

Wir rekonstruieren reverse Rechtsableitungen :-)

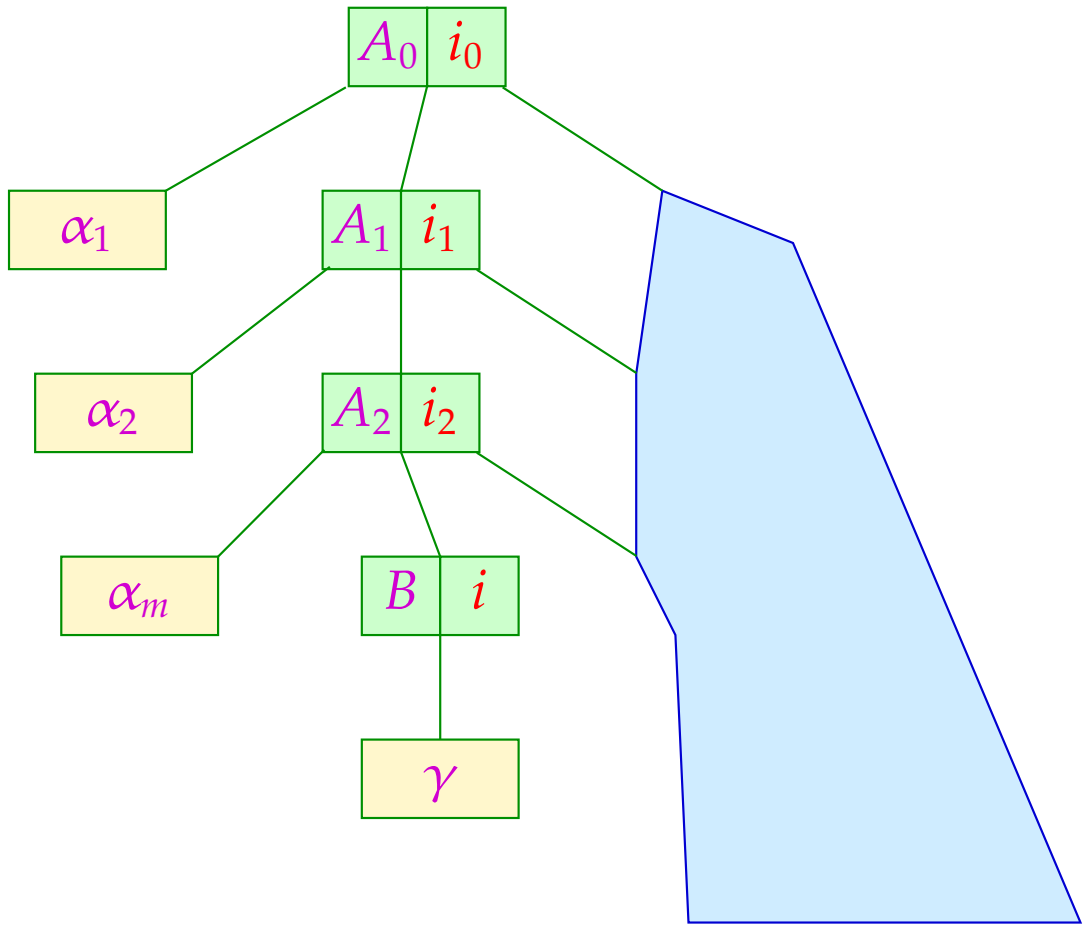
Dazu versuchen wir, für den Shift-Reduce-Parser $M_G^{(1)}$ die Reduktionsstellen zu identifizieren ...

Betrachte eine Berechnung dieses Kellerautomaten:

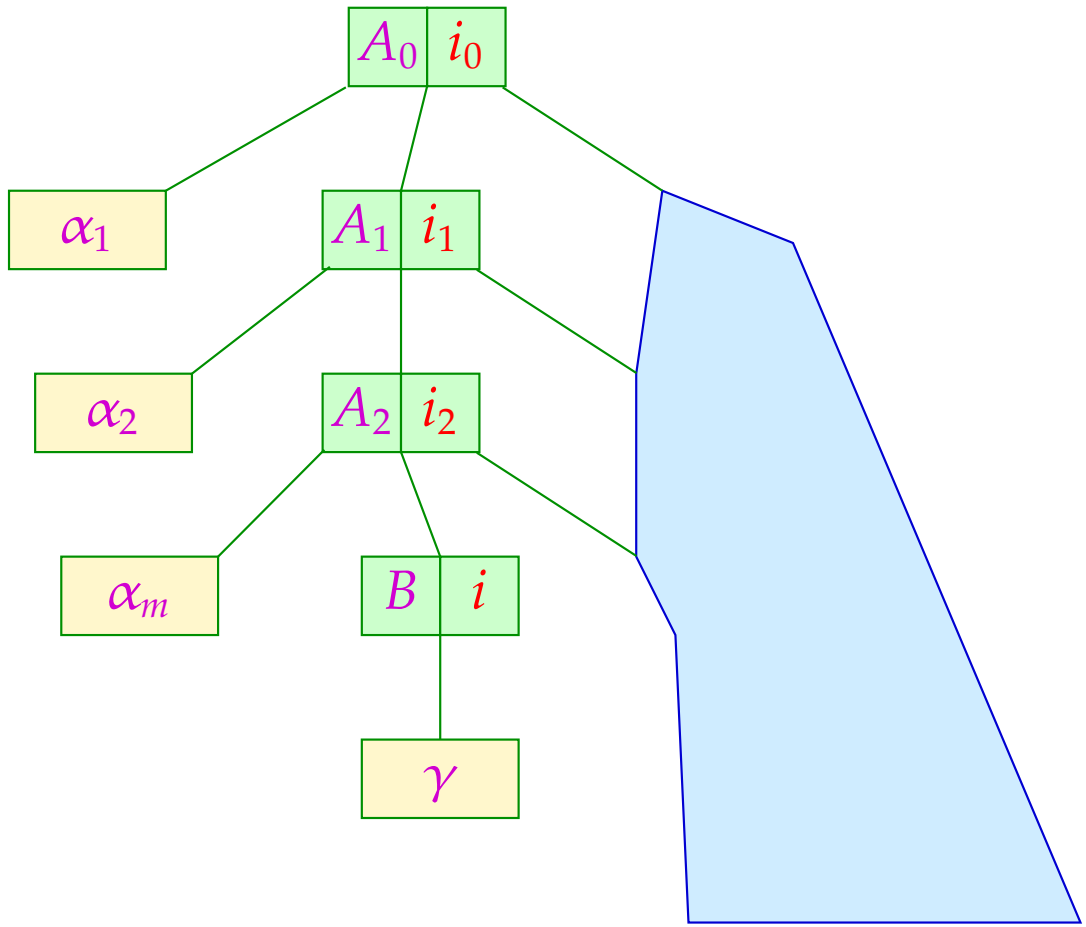
$$(q_0\,\alpha\,\gamma,\ v) \vdash (q_0\,\alpha\,B,\ v) \vdash^* (q_0\,S,\ \epsilon)$$

$\alpha\,\gamma$ nennen wir zuverlässiges Präfix für das vollständige Item $[B \to \gamma\bullet]$.

Dann ist $\alpha\,\gamma$ zuverlässig für $[B \to \gamma\bullet]$ gdw. $S \to_R^* \alpha\,B\,v$:-)



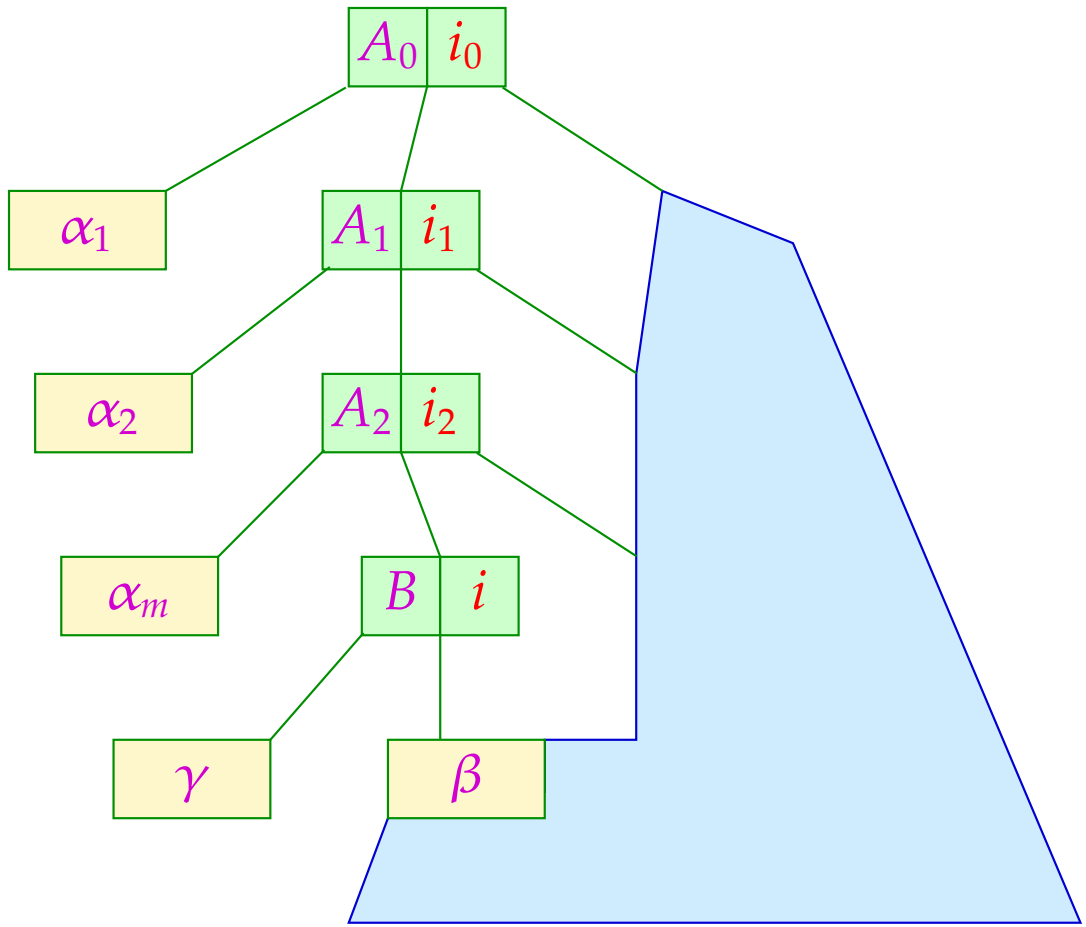
... wobei $\alpha = \alpha_1 \ldots \alpha_m$:-)



... wobei $\alpha = \alpha_1 \ldots \alpha_m$:-)

Umgekehrt können wir zu jedem möglichen Wort α' die Menge aller möglicherweise später passenden Regeln ermitteln ...

Das Item $[B \to \gamma \bullet \beta]$ heißt gültig für α' gdw. $S \to_R^* \alpha\, B\, v$ mit $\alpha' = \alpha\,\gamma$:



... wobei $\alpha = \alpha_1 \ldots \alpha_m$:-)

Beobachtung:

Die Menge der zuverlässigen Präfixe aus $(N \cup T)^*$ für (vollständige) Items kann mithilfe eines endlichen Automaten berechnet werden :-)

Zustände: Items :-)

Anfangszustand: $[S' \rightarrow \bullet\, S]$

Endzustände: $\{[B \rightarrow \gamma\bullet] \mid B \rightarrow \gamma \in P\}$

Übergänge:

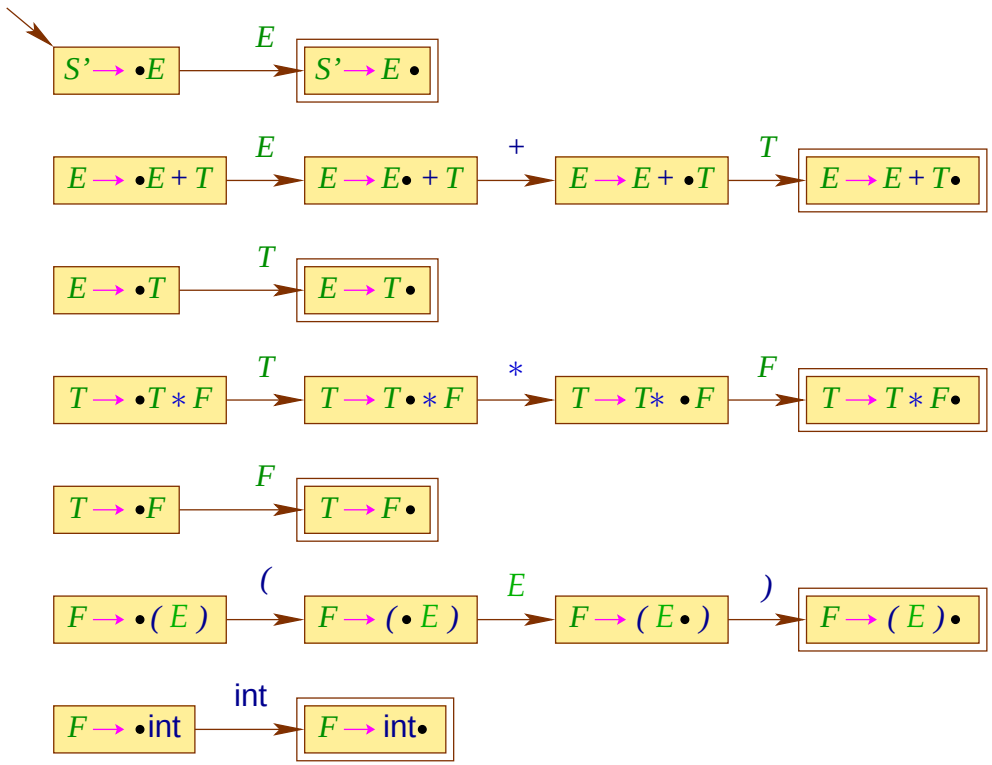
(1) $([A \rightarrow \alpha \bullet X\, \beta], X, [A \rightarrow \alpha\, X \bullet \beta]), \quad X \in (N \cup T), A \rightarrow \alpha\, X\, \beta \in P;$

(2) $([A \rightarrow \alpha \bullet B\, \beta], \epsilon, [B \rightarrow \bullet\, \gamma]), \quad A \rightarrow \alpha\, B\, \beta\,, \;\; B \rightarrow \gamma \in P;$

Den Automaten $c(G)$ nennen wir charakteristischen Automaten für G.

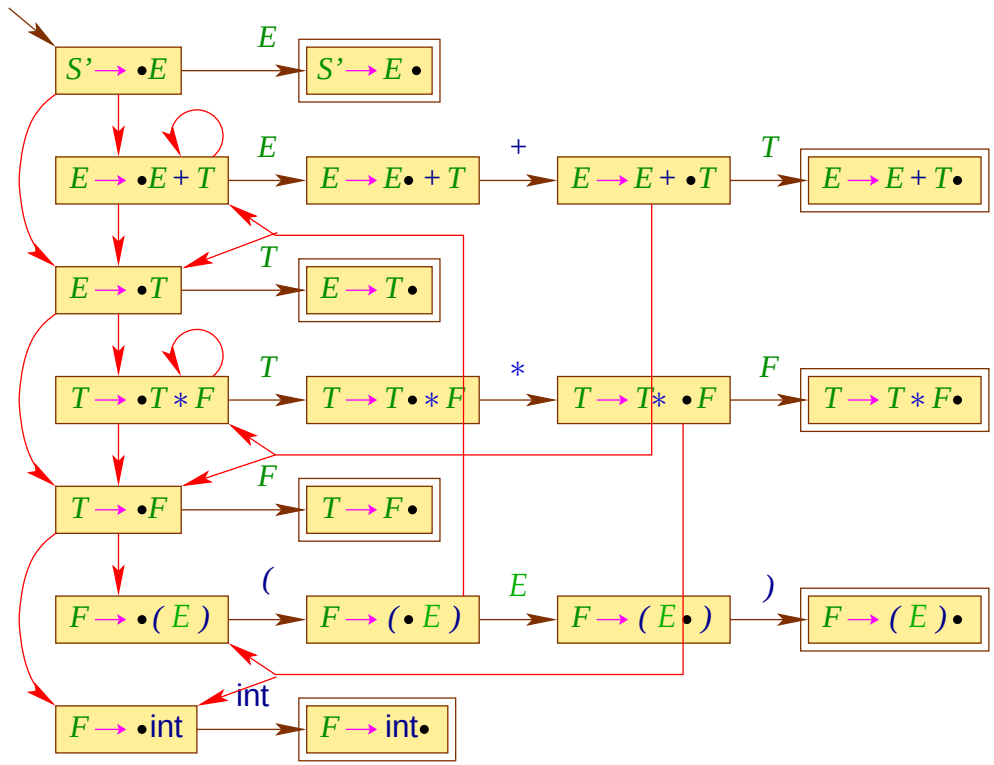
Beispiel:

$$
\begin{array}{lcll}
E & \rightarrow & E + T & \mid \; T \\
T & \rightarrow & T * F & \mid \; F \\
F & \rightarrow & (\; E \;) & \mid \; \mathsf{int}
\end{array}
$$



Beispiel:

$$
\begin{array}{lcll}
E & \rightarrow & E + T & \mid \quad T \\
T & \rightarrow & T * F & \mid \quad F \\
F & \rightarrow & (\; E \;) & \mid \quad \text{int}
\end{array}
$$

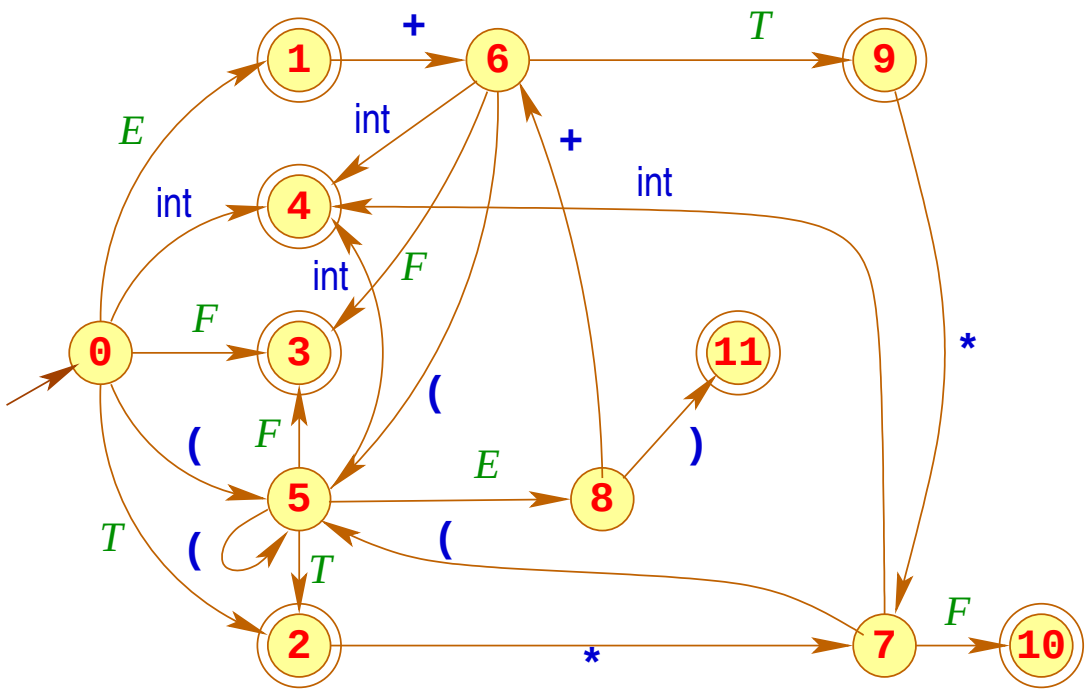


Den kanonischen $LR(0)$-Automaten $LR(G)$ erhalten wir aus $c(G)$, indem wir:

(1) nach jedem lesenden Übergang beliebig viele ϵ-Übergänge einschieben (unsere Konstruktion 1 zur Beseitigung von ϵ-Übergängen :-)

(2) die Teilmengenkonstruktion anwenden.

... im Beispiel:



Dazu konstruieren wir:

$$
\begin{aligned}
q_0 \;=\; \{&[S' \to \bullet\, E],\\
&[E \to \bullet\, E + T],\\
&[E \to \bullet\, T],\\
&[T \to \bullet\, T * F]\}\\
&[T \to \bullet\, F],\\
&[F \to \bullet\, (\ E\ )\ ],\\
&[F \to \bullet\, \mathsf{int}]\}
\end{aligned}
\qquad
\begin{aligned}
q_1 &= \delta(q_0, E) = \{[S' \to E\bullet],\\
&\qquad\qquad [E \to E \bullet + T]\}\\
q_2 &= \delta(q_0, T) = \{[E \to T\bullet],\\
&\qquad\qquad [T \to T \bullet * F]\}\\
q_3 &= \delta(q_0, F) = \{[T \to F\bullet]\}\\
q_4 &= \delta(q_0, \mathsf{int}) = \{[F \to \mathsf{int}\bullet]\}
\end{aligned}
$$

$$q_5 = \delta(q_0, ( ) = \{[F \to (\bullet E)],$$
$$[E \to \bullet E + T],$$
$$[E \to \bullet T],$$
$$[T \to \bullet T * F],$$
$$[T \to \bullet F],$$
$$[F \to \bullet (E)],$$
$$[F \to \bullet \mathsf{int}]\}$$

$$q_6 = \delta(q_1, +) = \{[E \to E + \bullet T],$$
$$[T \to \bullet T * F],$$
$$[T \to \bullet F],$$
$$[F \to \bullet (E)],$$
$$[F \to \bullet \mathsf{int}]\}$$

$$q_7 = \delta(q_2, *) = \{[T \to T * \bullet F],$$
$$[F \to \bullet (E)],$$
$$[F \to \bullet \mathsf{int}]\}$$

$$q_8 = \delta(q_5, E) = \{[F \to (E \bullet)]\}$$
$$[E \to E \bullet + T]\}$$

$$q_9 = \delta(q_6, T) = \{[E \to E + T \bullet],$$
$$[T \to T \bullet * F]\}$$

$$q_{10} = \delta(q_7, F) = \{[T \to T * F \bullet]\}$$

$$q_{11} = \delta(q_8,)) = \{[F \to (E) \bullet]\}$$

Beachte:

Der kanonische $LR(0)$-Automat kann auch direkt aus der Grammatik konstruiert werden :-)

Man benötigt die Hilfsfunktion:

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

Dann definiert man:

Zustände: Mengen von Items;

Anfangszustand: $\delta_\epsilon^* \{[S' \to \bullet\, S]\}$

Endzustände: $\{q \mid \exists\, A \to \alpha \in P : \quad [A \to \alpha \bullet] \in q\}$

Übergänge: $\delta(q, X) = \delta_\epsilon^* \{[A \to \alpha\, X \bullet\, \beta] \mid [A \to \alpha \bullet X\, \beta] \in q\}$

Idee zu einem Parser:

- Der Parser verwaltet ein zuverlässiges Präfix $\alpha = X_1 \dots X_m$ auf dem Keller und benutzt $LR(G)$, um Reduktionsstellen zu entdecken.
- Er kann mit einer Regel $A \to \gamma$ reduzieren, falls $[A \to \gamma \bullet]$ für α gültig ist :-)
- Damit der Automat nicht immer wieder neu über den Kellerinhalt laufen muss, kellern wir anstelle der X_i jeweils die Zustände !!!

Achtung:

Dieser Parser ist nur dann deterministisch, wenn jeder Endzustand des kanonischen $LR(0)$-Automaten keine Konflikte enthält ...

... im Beispiel:

$$
\begin{array}{lcl}
q_1 & = & \{[S' \to E\,\bullet], \\
 & & \quad [E \to E\,\bullet\,+\,T]\}
\end{array}
$$

$$
\begin{array}{lcl}
q_2 & = & \{[E \to T\,\bullet], \\
 & & \quad [T \to T\,\bullet * F]\}
\end{array}
\qquad
\begin{array}{lcl}
q_9 & = & \{[E \to E + T\,\bullet], \\
 & & \quad [T \to T\,\bullet * F]\}
\end{array}
$$

$$
\begin{array}{lcl}
q_3 & = & \{[T \to F\,\bullet]\}
\end{array}
\qquad
\begin{array}{lcl}
q_{10} & = & \{[T \to T\,*\,F\,\bullet]\}
\end{array}
$$

$$
\begin{array}{lcl}
q_4 & = & \{[F \to \mathsf{int}\,\bullet]\}
\end{array}
\qquad
\begin{array}{lcl}
q_{11} & = & \{[F \to (\,E\,)\,\bullet]\}
\end{array}
$$

Die Endzustände q_1, q_2, q_9 enthalten mehr als ein Item :-(

Aber wir haben ja auch noch nicht Vorausschau eingesetzt :-)

Die Konstruktion des $LR(0)$-Parsers:

Zustände: $Q \cup \{f\}$ (f neu :-)

Anfangszustand: q_0

Endzustand: f

Übergänge:

Shift:	$(p, a, p\,q)$	falls	$q = \delta(p, a) \neq \emptyset$
Reduce:	$(p\,q_1 \ldots q_m, \epsilon, p\,q)$	falls	$[A \to X_1 \ldots X_m \bullet] \in q_m$, $q = \delta(p, A)$
Finish:	$(q_0\,p, \epsilon, f)$	falls	$[S' \to S\bullet] \in p$

wobei $LR(G) = (Q, T, \delta, q_0, F)$.

Zur Korrektheit:

Man zeigt:

Die akzeptierenden Berechnungen des $LR(0)$-Parsers stehen in eins-zu-eins Beziehung zu denen des Shift-Reduce-Parsers $M_G^{(1)}$.

Wir folgern:

$\Longrightarrow$ Die akzeptierte Sprache ist genau $\mathcal{L}(G)$:-)

$\Longrightarrow$ Die Folge der Reduktionen einer akzeptierenden Berechnung für ein Wort $w \in T$ liefert eine reverse Rechts-Ableitung von G für w :-)

Leider ist der $LR(0)$-Parser i.a. nicht-deterministisch :-(

Wir identifizieren zwei Gründe:

Reduce-Reduce-Konflikt:

$$[A \to \gamma \bullet], \; [A' \to \gamma' \bullet] \in q \quad \text{mit} \quad A \neq A' \vee \gamma \neq \gamma'$$

Shift-Reduce-Konflikt:

$$[A \to \gamma \bullet], \; [A' \to \alpha \bullet a\,\beta] \in q \quad \text{mit} \quad a \in T$$

für einen Zustand $q \in Q$.

Solche Zustände nennen wir ungeeignet.

Idee:

Benutze k-Vorausschau, um Konflikte zu lösen.

Wir definieren:

Die reduzierte kontextfreie Grammatik G heißt $LR(k)$-Grammatik, falls für $\mathsf{First}_k(w) = \mathsf{First}_k(x)$ aus:

$$\left.\begin{array}{lllll} S & \to_R^* & \alpha\, A\, w & \to & \alpha\, \beta\, w \\ S & \to_R^* & \alpha'\, A'\, w' & \to & \alpha\, \beta\, x \end{array}\right\} \quad \text{folgt:} \quad \alpha = \alpha' \wedge A = A' \wedge w' = x$$

Beispiele:

(1) $S \to A \mid B \qquad A \to a\,A\,b \mid 0 \qquad B \to a\,B\,b\,b \mid 1$

... ist nicht $LL(k)$ für jedes k — aber $LR(0)$:

Sei $S \to_R^* \alpha\,X\,w \to \alpha\,\beta\,w$. Dann ist $\alpha\,\underline{\beta}$ von einer der Formen:

$$\underline{A}\ ,\ \underline{B}\ ,\ a^n\,\underline{a\,A\,b}\ ,\ a^n\,\underline{a\,B\,b\,b}\ ,\ a^n\,\underline{0}\ ,\ a^n\,\underline{1} \qquad (n \geq 0)$$

Beispiele:

(1) $S \to A \mid B \qquad A \to a\,A\,b \mid 0 \qquad B \to a\,B\,b\,b \mid 1$

... ist nicht $LL(k)$ für jedes k — aber $LR(0)$:

Sei $S \to_R^* \alpha\,X\,w \to \alpha\,\beta\,w$. Dann ist $\alpha\,\underline{\beta}$ von einer der Formen:

$$\underline{A}\ ,\ \underline{B}\ ,\ a^n\,\underline{a\,A\,b}\ ,\ a^n\,\underline{a\,B\,b\,b}\ ,\ a^n\,\underline{0}\ ,\ a^n\,\underline{1} \qquad (n \geq 0)$$

(2) $S \to a\,A\,c \qquad A \to A\,b\,b \mid b$

... ist ebenfalls $LR(0)$:

Sei $S \to_R^* \alpha\,X\,w \to \alpha\,\beta\,w$. Dann ist $\alpha\,\underline{\beta}$ von einer der Formen:

$$a\,\underline{b}\ ,\ a\,\underline{A\,b\,b}\ ,\ \underline{a\,A\,c}$$

(3) $S \to a\,A\,c \qquad A \to b\,b\,A \mid b$ ... ist nicht $LR(0)$, aber $LR(1)$:

Für $S \to_R^* \alpha\,X\,w \to \alpha\,\beta\,w$ mit $\{y\} = \mathsf{First}_k(w)$ ist $\alpha\,\underline{\beta}\,y$ von einer der Formen:

$$a\,b^{2n}\,\underline{b\,c}\;,\; a\,b^{2n}\,\underline{b\,b\,A\,c}\;,\; \underline{a\,A\,c}$$