2.5 Schnelle Berechnung von Vorausschau-Mengen

Im Fall $k = 1$ lassen sich First, Follow besonders effizient berechnen ;-)

Beobachtung:

Seien $L_1, L_2 \subseteq T \cup \{\epsilon\}$ mit $L_1 \neq \emptyset \neq L_2$. Dann ist:

$$L_1 \odot L_2 = \begin{cases} L_1 & \text{falls} \quad \epsilon \notin L_1 \\ (L_1 \setminus \{\epsilon\}) \cup L_2 & \text{sonst} \end{cases}$$

Ist G reduziert, sind alle Mengen $\mathsf{First}_1(A)$ nichtleer :-)

Idee:

- Behandle ϵ separat!

  Sei $\mathsf{empty}(X) = \mathsf{true}$ gdw. $X \to^* \epsilon$.

- Definiere die ϵ-freien First_1-Mengen

$$\begin{array}{lcll} F_\epsilon(a) & = & \{a\} & \text{für } a \in T \\ F_\epsilon(A) & = & \mathsf{First}_1(A) \backslash \{\epsilon\} & \text{für } A \in N \end{array}$$

- Konstruiere direkt ein Ungleichungssystem für $F_\epsilon(A)$:

$$F_\epsilon(A) \supseteq F_\epsilon(X_j) \quad \text{falls} \quad A \to X_1 \ldots X_m \in P, \; \mathsf{empty}(X_1) \wedge \ldots \wedge \mathsf{empty}(X_{j-1})$$

... im Beispiel:

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

wobei $\mathsf{empty}(E) = \mathsf{empty}(T) = \mathsf{empty}(F) = \mathsf{false}$.

Deshalb erhalten wir:

$$\begin{array}{lcllcl} F_\epsilon(S') & \supseteq & F_\epsilon(E) & F_\epsilon(E) & \supseteq & F_\epsilon(E) \\ F_\epsilon(E) & \supseteq & F_\epsilon(T) & F_\epsilon(T) & \supseteq & F_\epsilon(T) \\ F_\epsilon(T) & \supseteq & F_\epsilon(F) & F_\epsilon(F) & \supseteq & \{ \, (, \mathsf{name}, \mathsf{int} \} \end{array}$$

Entsprechend konstruieren wir zur Berechnung von Follow_1 :

$$
\begin{array}{lcll}
\mathsf{Follow}_1(S) & \supseteq & \{\epsilon\} & \\
\mathsf{Follow}_1(B) & \supseteq & F_\epsilon(X_j) & \text{falls} \quad A \to \alpha\, B\, X_1 \ldots X_m \in P, \\
 & & & \mathsf{empty}(X_1) \wedge \ldots \wedge \mathsf{empty}(X_{j-1}) \\
\mathsf{Follow}_1(B) & \supseteq & \mathsf{Follow}_1(A) & \text{falls} \quad A \to \alpha\, B\, X_1 \ldots X_m \in P, \\
 & & & \mathsf{empty}(X_1) \wedge \ldots \wedge \mathsf{empty}(X_m)
\end{array}
$$

Entsprechend konstruieren wir zur Berechnung von Follow_1 :

$$\begin{array}{lcll}
\mathsf{Follow}_1(S) & \supseteq & \{\epsilon\} & \\
\mathsf{Follow}_1(B) & \supseteq & F_\epsilon(X_j) & \text{falls} \quad A \to \alpha\, B\, X_1 \ldots X_m \in P, \\
 & & & \mathsf{empty}(X_1) \wedge \ldots \wedge \mathsf{empty}(X_{j-1}) \\
\mathsf{Follow}_1(B) & \supseteq & \mathsf{Follow}_1(A) & \text{falls} \quad A \to \alpha\, B\, X_1 \ldots X_m \in P, \\
 & & & \mathsf{empty}(X_1) \wedge \ldots \wedge \mathsf{empty}(X_m)
\end{array}$$

... im Beispiel:

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

Entsprechend konstruieren wir zur Berechnung von Follow_1 :

$$\begin{array}{lcll}
\mathsf{Follow}_1(S) & \supseteq & \{\epsilon\} & \\
\mathsf{Follow}_1(B) & \supseteq & F_\epsilon(X_j) & \text{falls} \quad A \to \alpha\, B\, X_1 \ldots X_m \in P, \\
 & & & \mathsf{empty}(X_1) \wedge \ldots \wedge \mathsf{empty}(X_{j-1}) \\
\mathsf{Follow}_1(B) & \supseteq & \mathsf{Follow}_1(A) & \text{falls} \quad A \to \alpha\, B\, X_1 \ldots X_m \in P, \\
 & & & \mathsf{empty}(X_1) \wedge \ldots \wedge \mathsf{empty}(X_m)
\end{array}$$

... im Beispiel:

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

... erhalten wir:

$$\begin{array}{lcllcl}
\mathsf{Follow}_1(S') & \supseteq & \{\epsilon\} & \mathsf{Follow}_1(E) & \supseteq & \mathsf{Follow}_1(S') \\
\mathsf{Follow}_1(E) & \supseteq & \{+,)\} & \mathsf{Follow}_1(T) & \supseteq & \{*\} \\
\mathsf{Follow}_1(T) & \supseteq & \mathsf{Follow}_1(E) & \mathsf{Follow}_1(F) & \supseteq & \mathsf{Follow}_1(T)
\end{array}$$

Diskussion:

- Diese Ungleichungssysteme bestehen aus Ungleichungen der Form:

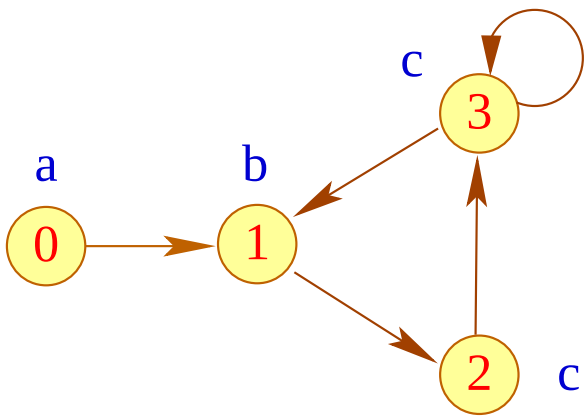
$$x \sqsupseteq y \qquad \text{bzw.} \qquad x \sqsupseteq d$$

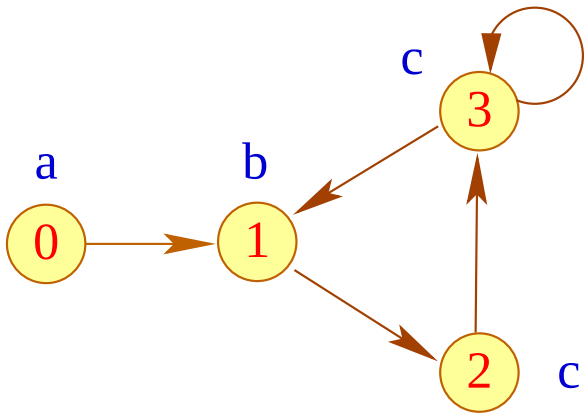
  für Variablen x, y und $d \in \mathbb{D}$.

- Solche Ungleichungssysteme heißen reine Vereinigungs-Probleme :-)
- Diese Probleme können mit linearem Aufwand gelöst werden ...

Beispiel: $\mathbb{D} = 2^{\{a,b,c\}}$

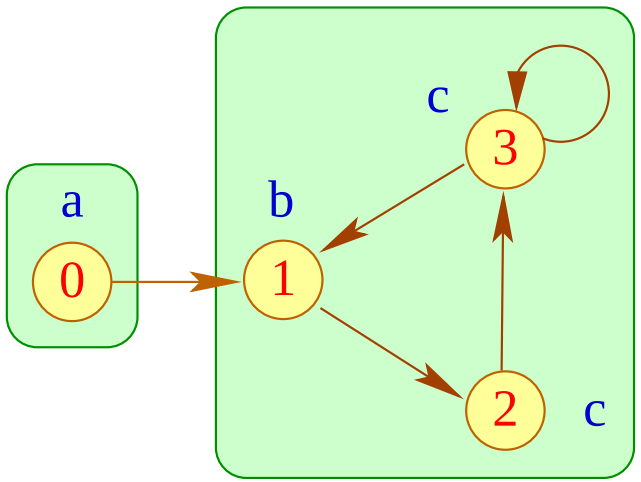
$$\begin{array}{lll} x_0 \supseteq \{a\} & & \\ x_1 \supseteq \{b\} & x_1 \supseteq x_0 & x_1 \supseteq x_3 \\ x_2 \supseteq \{c\} & x_2 \supseteq x_1 & \\ x_3 \supseteq \{c\} & x_3 \supseteq x_2 & x_3 \supseteq x_3 \end{array}$$





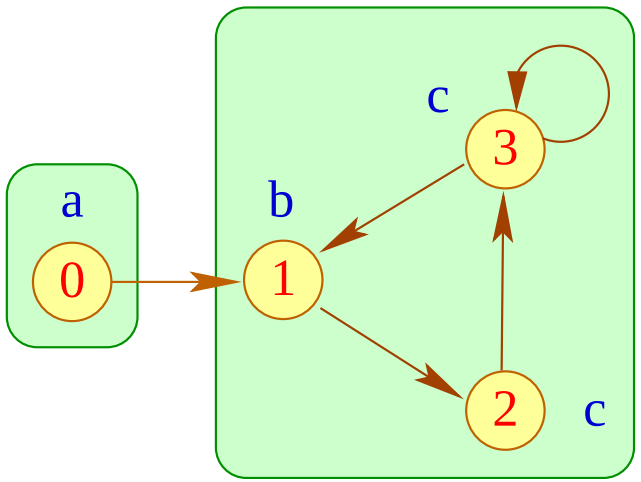
Vorgehen:

- Konstruiere den Variablen-Abhängigkeitsgraph zum Ungleichungssystem.



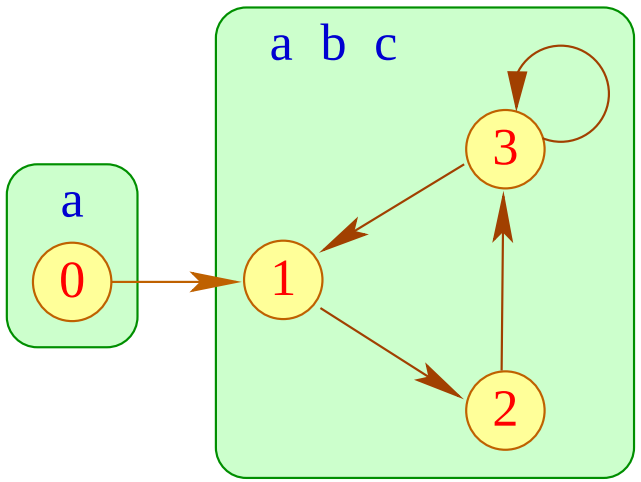
Vorgehen:

- Konstruiere den Variablen-Abhängigkeitsgraph zum Ungleichungssystem.
- Innerhalb einer starken Zusammenhangskomponente haben alle Variablen den gleichen Wert :-)



Vorgehen:

- Konstruiere den Variablen-Abhängigkeitsgraph zum Ungleichungssystem.
- Innerhalb einer starken Zusammenhangskomponente haben alle Variablen den gleichen Wert :-)
- Hat eine SZK keine eingehenden Kanten, erhält man ihren Wert, indem man die kleinste obere Schranke aller Werte in der SZK berechnet :-)

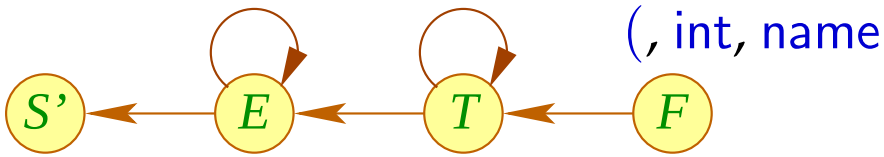


Vorgehen:

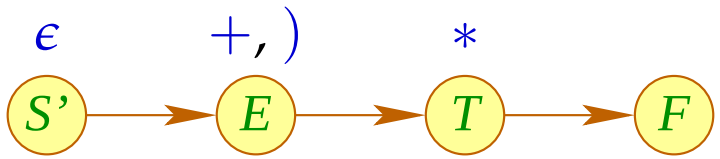
- Konstruiere den Variablen-Abhängigkeitsgraph zum Ungleichungssystem.
- Innerhalb einer starken Zusammenhangskomponente haben alle Variablen den gleichen Wert :-)
- Hat eine SZK keine eingehenden Kanten, erhält man ihren Wert, indem man die kleinste obere Schranke aller Werte in der SZK berechnet :-)
- Gibt es eingehende Kanten, muss man zusätzlich die Werte an deren Startknoten hinzufügen :-)

... für unsere Beispiel-Grammatik:

First_1 :



Follow_1 :



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 :-)

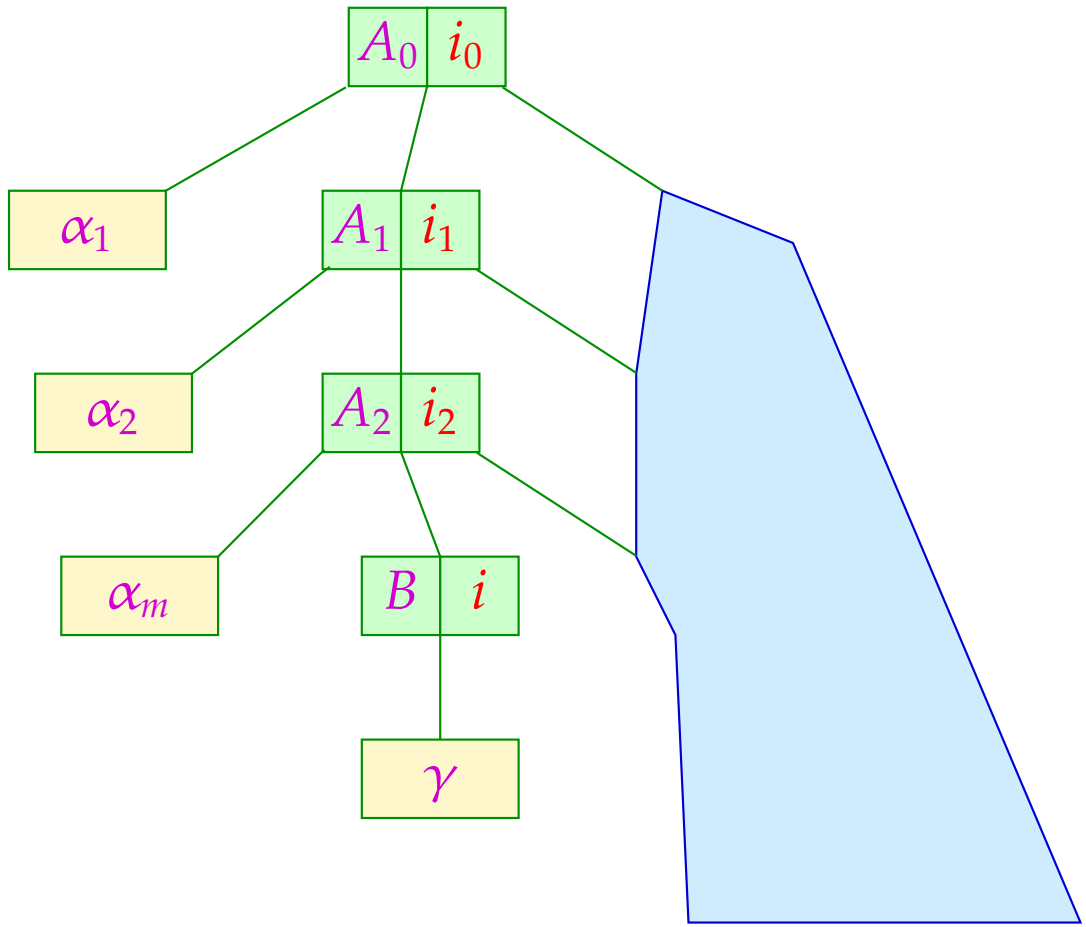
Dazu versuchen wir, für den Shift-Reduce-Parser $M_G^{(1)}$ die Reduktionsstellen zu identifizieren ...

Betrachte eine Berechnung dieses Kellerautomaten:

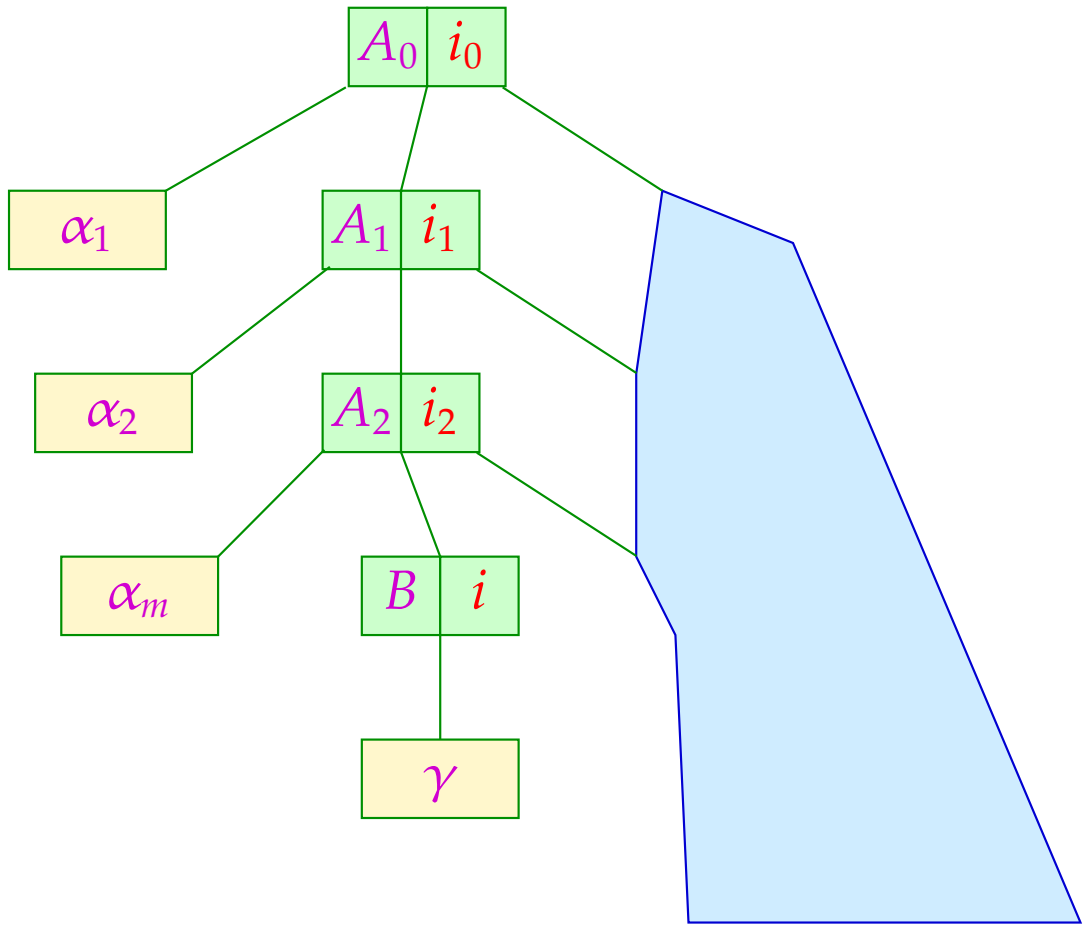
$$(q_0\,\alpha\,\gamma,\ v) \vdash (q_0\,\alpha\,B,\ v) \vdash^* (q_0\,S,\ \epsilon)$$

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

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



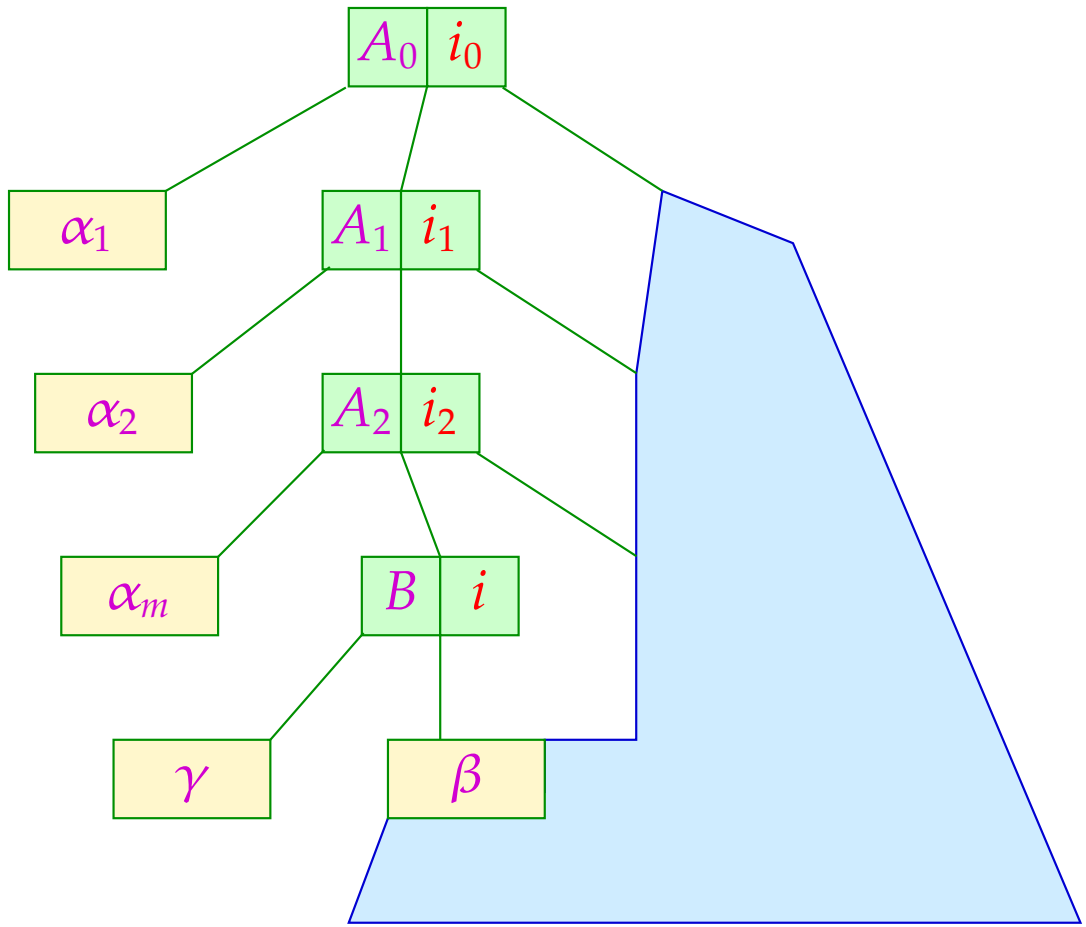
... wobei $\alpha = \alpha_1 \ldots \alpha_m$:-)



... 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: $[S' \to \bullet\, S]$

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

Übergänge:

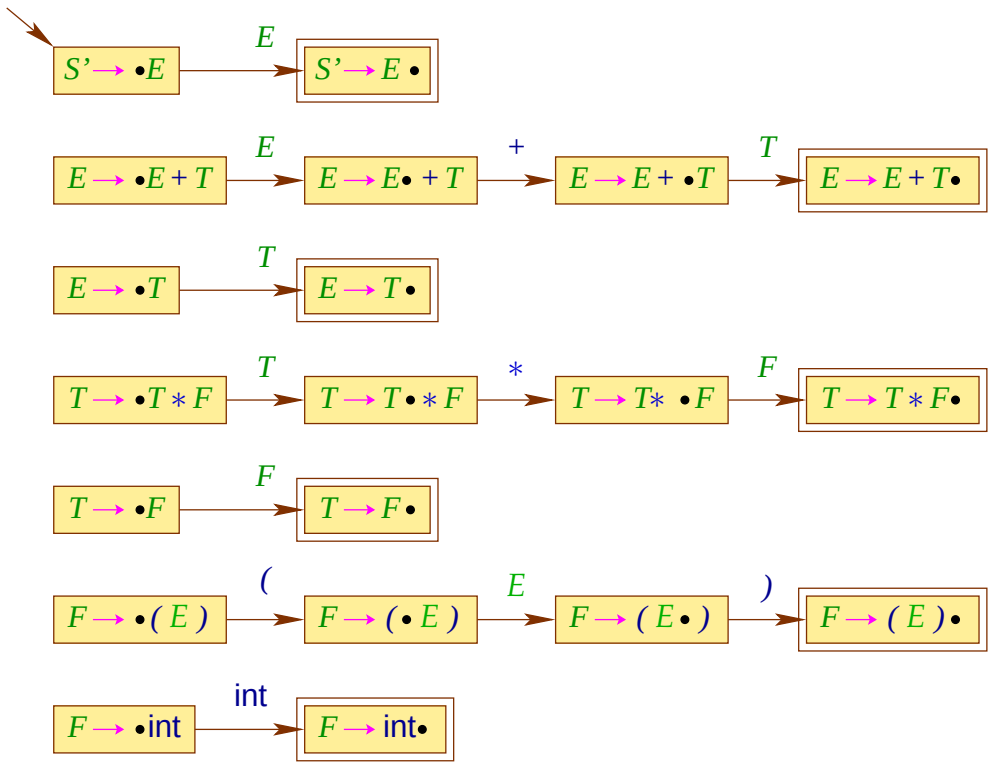
(1) $([A \to \alpha \bullet X\, \beta], X, [A \to \alpha\, X \bullet \beta]), \quad X \in (N \cup T), A \to \alpha\, X\, \beta \in P;$

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

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

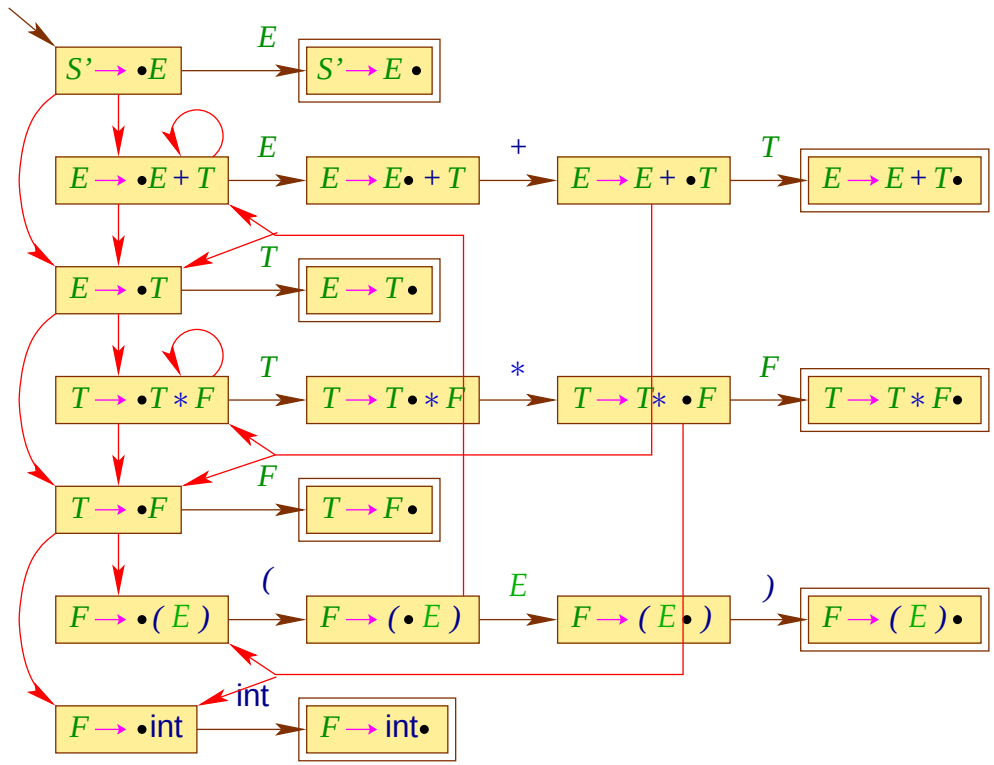
Beispiel:

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



Beispiel:

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

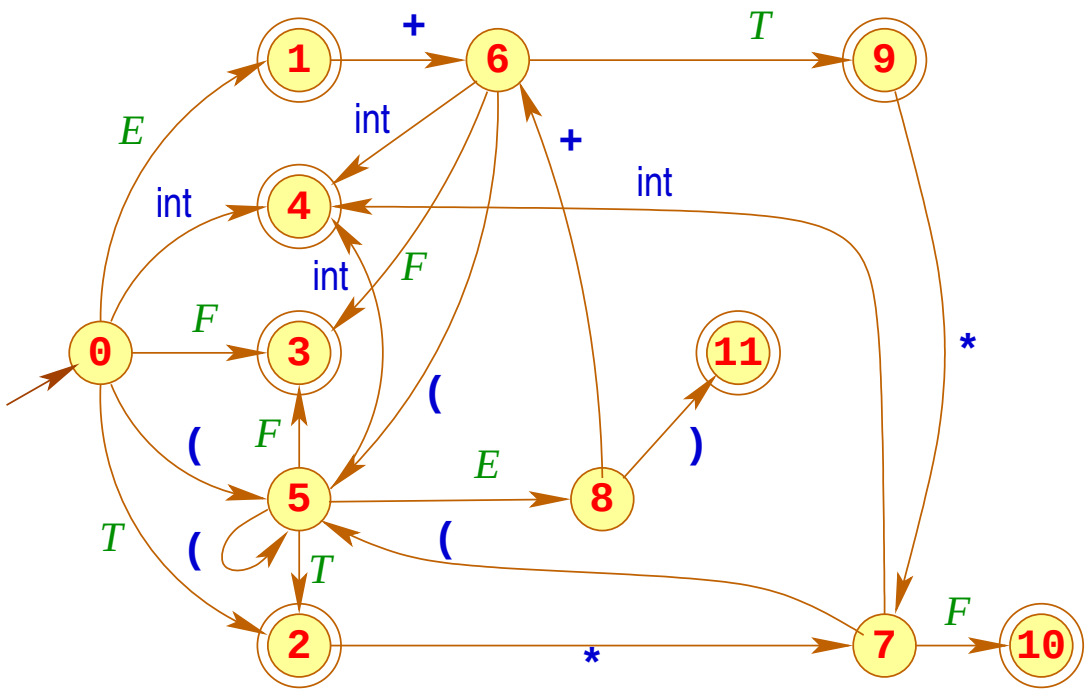


Den kanonischen $LR(0)$-Automaten $LR(G)$ erhalten wir aus $c(G)$, indem wir:

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