Satz

Eine reduzierte kontextfreie Grammatik G ist genau dann $LR(k)$ wenn der kanonische $LR(k)$-Automat $LR(G, k)$ keine $LR(k)$-ungeeigneten Zustände besitzt.

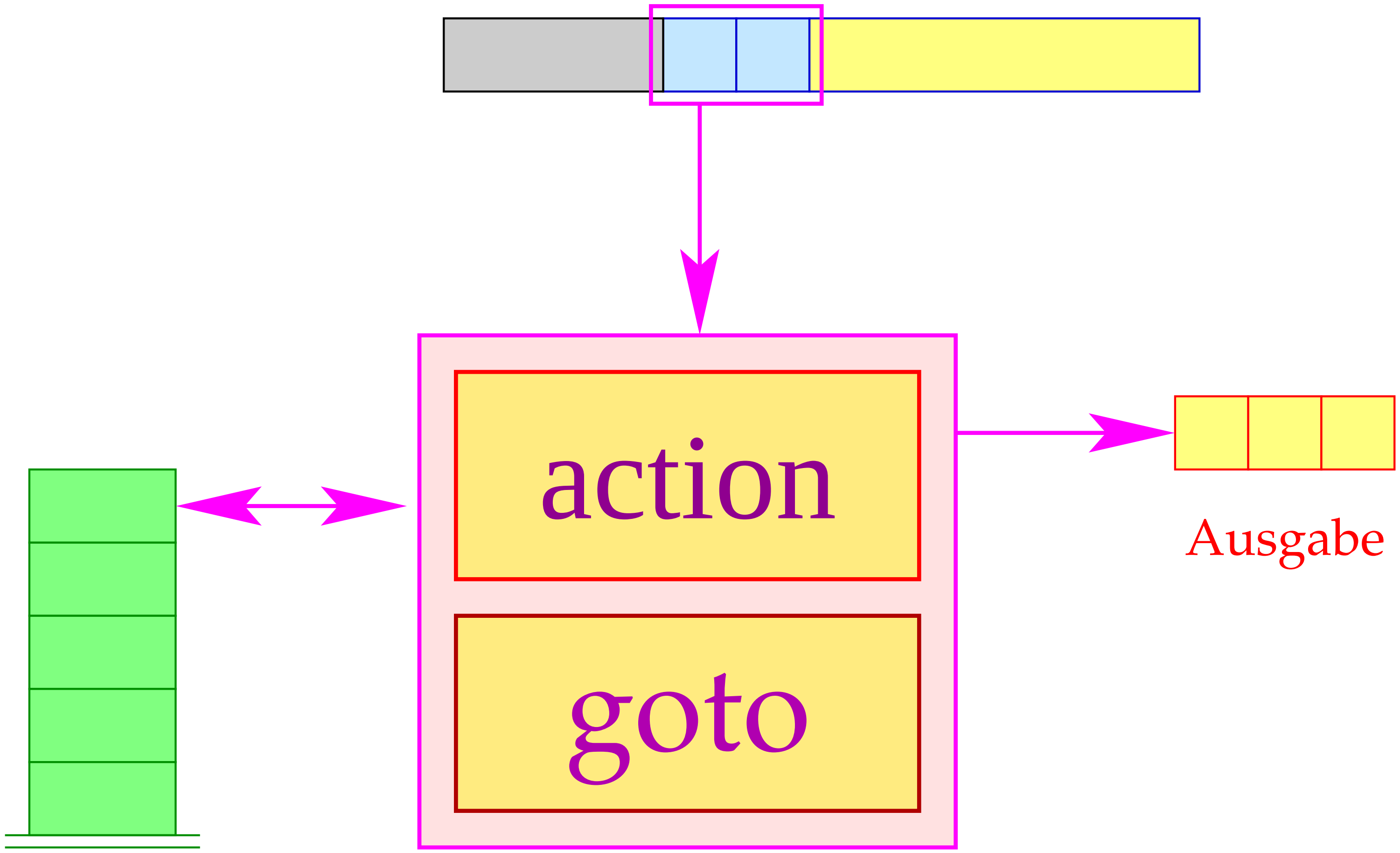
Diskussion:

- Unser Beispiel ist offenbar $LR(1)$:-)
- Im Allgemeinen hat der kanonische $LR(k)$-Automat sehr viel mehr Zustände als $LR(G) = LR(G, 0)$:-(
- Man betrachtet darum i.a. Teilklassen von $LR(k)$-Grammatiken, bei denen man nur $LR(G)$ benutzt ...
- Zur Konflikt-Auflösung ordnet man den Items in den Zuständen Vorausschau-Mengen zu:

  (1) Die Zuordnung ist unabhängig vom Zustand $\Longrightarrow$ Simple $LR(k)$

  (2) Die Zuordnung hängt vom Zustand ab $\Longrightarrow$ $LALR(k)$

Der $LR(k)$-Parser:



Erläuterung:

- Die goto-Tabelle kodiert die Zustandsübergänge:

$$\text{goto}[q, X] = \delta(q, X) \quad \in \quad Q$$

- Die action-Tabelle beschreibt für jeden Zustand q und möglichen Look-ahead w die erforderliche Aktion.

Diese sind:

shift	//	Shift-Operation
reduce $(A \rightarrow \gamma)$	//	Reduktion mit Ausgabe
error	//	Fehler

... im Beispiel:

$$
\begin{array}{lcl}
E & \rightarrow & E + T^{\,0} \quad | \quad T^{\,1} \\
T & \rightarrow & T * F^{\,0} \quad | \quad F^{\,1} \\
F & \rightarrow & (\ E\ )^{\,0} \quad | \quad \mathsf{int}^{\,1}
\end{array}
$$

action	ϵ	int	(	)	$+$	$*$
q_1	$S', 0$					s
q_2	$E, 1$					s
q'_2				$E, 1$		s
q_3	$T, 1$				$T, 1$	$T, 1$
q'_3				$T, 1$	$T, 1$	$T, 1$
q_4	$F, 1$				$F, 1$	$F, 1$
q'_4				$F, 1$	$F, 1$	$F, 1$
q_9	$E, 0$				$E, 0$	s
q'_9				$E, 0$	$E, 0$	s
q_{10}	$T, 0$				$T, 0$	$T, 0$
q'_{10}				$T, 0$	$T, 0$	$T, 0$
q_{11}	$F, 0$				$F, 0$	$F, 0$
q'_{11}				$F, 0$	$F, 0$	$F, 0$

2.7 Spezielle Bottom-up-Verfahren mit $LR(G)$

Idee 1: Benutze $\textsf{Follow}_k$-Mengen zur Konflikt-Lösung ...

Reduce-Reduce-Konflikt:

Falls für $[A \to \gamma \bullet], \; [A' \to \gamma' \bullet] \; \in \; q$ mit $A \neq A' \vee \gamma \neq \gamma'$,

$$\textsf{Follow}_k(A) \cap \textsf{Follow}_k(A') \neq \emptyset$$

Shift-Reduce-Konflikt:

Falls für $[A \to \gamma \bullet], \; [A' \to \alpha \bullet a\, \beta] \; \in \; q$ mit $a \in T$,

$$\textsf{Follow}_k(A) \cap (\{a\} \odot \textsf{First}_k(\beta) \odot \textsf{Follow}_k(A')) \neq \emptyset$$

für einen Zustand $q \in Q$.

Dann nennen wir den Zustand q $SLR(k)$-ungeeignet :-)

Die reduzierte Grammatik G nennen wir $SLR(k)$ (simple $LR(k)$:-), falls der kanonische $LR(0)$-Automat $LR(G)$ keine $SLR(k)$-ungeeigneten Zustände enthält :-)

Die reduzierte Grammatik G nennen wir $SLR(k)$ (simple $LR(k)$:-), falls der kanonische $LR(0)$-Automat $LR(G)$ keine $SLR(k)$-ungeeigneten Zustände enthält :-)

... im Beispiel:

Bei unserer Beispiel-Grammatik treten Konflikte möglicherweise in den Zuständen q_1, q_2, q_9 auf:

$$
\begin{array}{lll}
q_1 & = & \{[S' \to E\bullet], \\
 & & \;[E \to E \bullet + T]\}
\end{array}
\qquad
\begin{array}{lll}
\mathsf{Follow}_1(S') \cap \{+\} \odot \{\ldots\} & = & \{\epsilon\} \cap \{+\} \\
 & = & \emptyset
\end{array}
$$

Die reduzierte Grammatik G nennen wir $SLR(k)$ (simple $LR(k)$:-), falls der kanonische $LR(0)$-Automat $LR(G)$ keine $SLR(k)$-ungeeigneten Zustände enthält :-)

... im Beispiel:

Bei unserer Beispiel-Grammatik treten Konflikte möglicherweise in den Zuständen q_1, q_2, q_9 auf:

$$
\begin{array}{lll}
q_1 & = & \{[S' \to E\bullet], \\
 & & \;[E \to E \bullet + T]\}
\end{array}
\qquad
\begin{array}{rcl}
\mathsf{Follow}_1(S') \cap \{+\} \odot \{\dots\} & = & \{\epsilon\} \cap \{+\} \\
 & = & \emptyset
\end{array}
$$

$$
\begin{array}{lll}
q_2 & = & \{[E \to T\bullet], \\
 & & \;[T \to T \bullet * F]\}
\end{array}
\qquad
\begin{array}{rcl}
\mathsf{Follow}_1(E) \cap \{*\} \odot \{\dots\} & = & \{\epsilon, +,)\} \cap \{*\} \\
 & = & \emptyset
\end{array}
$$

$$
\begin{array}{lll}
q_9 & = & \{[E \to E + T\bullet], \\
 & & \;[T \to T \bullet * F]\}
\end{array}
\qquad
\begin{array}{rcl}
\mathsf{Follow}_1(E) \cap \{*\} \odot \{\dots\} & = & \{\epsilon, +,)\} \cap \{*\} \\
 & = & \emptyset
\end{array}
$$

Idee 2: Berechne für jeden Zustand q Follow-Mengen :-)

Für $[A \to \alpha \bullet \beta] \in q$ definieren wir:

$$\begin{array}{lll} \Lambda_k(q, [A \to \alpha \bullet \beta]) & = & \{\mathsf{First}_k(w) \mid S' \to_R^* \gamma\, A\, w \;\; \wedge \\ & & \qquad\qquad \delta(q_0, \gamma\,\alpha) = q\} \\ & // & \subseteq \;\; \mathsf{Follow}_k(A) \end{array}$$

Idee 2: Berechne für jeden Zustand q Follow-Mengen :-)

Für $[A \to \alpha \bullet \beta] \in q$ definieren wir:

$$\begin{array}{lll} \Lambda_k(q, [A \to \alpha \bullet \beta]) & = & \{\text{First}_k(w) \mid S' \to_R^* \gamma\, A\, w \;\wedge \\ & & \qquad\qquad \delta(q_0, \gamma\, \alpha) = q\} \\ & // \subseteq & \text{Follow}_k(A) \end{array}$$

Reduce-Reduce-Konflikt:

$[A \to \gamma \bullet]\,,\; [A' \to \gamma' \bullet] \in q$ mit $A \neq A' \vee \gamma \neq \gamma'$ wobei:

$$\Lambda_k(q, [A \to \gamma \bullet]) \cap \Lambda_k(q, [A' \to \gamma' \bullet]) \neq \emptyset$$

Idee 2: Berechne für jeden Zustand q Follow-Mengen :-)

Für $[A \to \alpha \bullet \beta] \in q$ definieren wir:

$$\begin{array}{lll} \Lambda_k(q, [A \to \alpha \bullet \beta]) & = & \{\mathsf{First}_k(w) \mid S' \to_R^* \gamma\, A\, w \;\wedge \\ & & \qquad\qquad \delta(q_0, \gamma\,\alpha) = q\} \\ & // & \subseteq \; \mathsf{Follow}_k(A) \end{array}$$

Reduce-Reduce-Konflikt:

$[A \to \gamma \bullet]\,,\;\; [A' \to \gamma' \bullet] \;\in\; q$ mit $A \neq A' \vee \gamma \neq \gamma'$ wobei:

$$\Lambda_k(q, [A \to \gamma \bullet]) \;\cap\; \Lambda_k(q, [A' \to \gamma' \bullet]) \;\neq\; \emptyset$$

Shift-Reduce-Konflikt:

$[A \to \gamma \bullet]\,,\;\; [A' \to \alpha \bullet a\, \beta] \;\in\; q$ mit $a \in T$ wobei:

$$\Lambda_k(q, [A \to \gamma \bullet]) \;\cap\; (\{a\} \odot \mathsf{First}_k(\beta) \odot \Lambda_k(q, [A' \to \alpha \bullet a\, \beta])) \;\neq\; \emptyset$$

Solche Zustände nennen wir jetzt $LALR(k)$-ungeeignet :-)

Die reduzierte Grammatik G nennen wir $LALR(k)$, falls der kanonische $LR(0)$-Automat $LR(G)$ keine $LALR(k)$-ungeeigneten Zustände enthält :-)

Bevor wir Beispiele betrachten, überlegen wir erst, wie die Mengen $\Lambda_k(q, [A \to \alpha \bullet \beta])$ berechnet werden können :-)

Die reduzierte Grammatik G nennen wir $LALR(k)$, falls der kanonische $LR(0)$-Automat $LR(G)$ keine $LALR(k)$-ungeeigneten Zustände enthält :-)

Bevor wir Beispiele betrachten, überlegen wir erst, wie die Mengen $\Lambda_k(q, [A \to \alpha \bullet \beta])$ berechnet werden können :-)

Idee: Stelle ein Ungleichungssystem auf !!!

$$
\begin{array}{lll}
\Lambda_k(q_0, [S' \to \bullet\, S]) & \supseteq & \{\epsilon\} \\
\Lambda_k(q, [A \to \alpha\, X \bullet \beta]) & \supseteq & \Lambda_k(p, [A \to \alpha \bullet X\, \beta]) \qquad \text{falls} \quad \delta(p, X) = q \\
\Lambda_k(q, [A \to \bullet\, \gamma]) & \supseteq & \mathsf{First}_k(\beta) \odot \Lambda_k(q, [B \to \alpha \bullet A\, \beta]) \qquad \text{falls} \quad [B \to \alpha \bullet A\, \beta] \in q
\end{array}
$$

Beispiel:

$$\begin{array}{lll} S & \rightarrow & A\,b\,B \mid B \\ A & \rightarrow & a \mid b\,B \\ B & \rightarrow & A \end{array}$$

Der kanonische $LR(0)$-Automat hat dann die folgenden Zustände:

$$\begin{array}{lll} q_0 & = & \{[S' \rightarrow \bullet\, S], \\ & & \;[S \rightarrow \bullet\, A\,b\,B], \\ & & \;[A \rightarrow \bullet\, a], \\ & & \;[A \rightarrow \bullet\, b\,B], \\ & & \;[S \rightarrow \bullet\, B], \\ & & \;[B \rightarrow \bullet\, A]\} \end{array}$$

$$q_1 = \delta(q_0, S) = \{[S' \rightarrow S\bullet]\}$$

$$q_2 = \delta(q_0, a) = \{[A \rightarrow a\bullet]\}$$

$$\begin{array}{lllll} q_3 & = & \delta(q_0, b) & = & \{[A \rightarrow b \bullet B], \\ & & & & \;[B \rightarrow \bullet\, A], \\ & & & & \;[A \rightarrow \bullet\, a], \\ & & & & \;[A \rightarrow \bullet\, b\,B]\} \end{array}$$

$$q_4 = \delta(q_0, B) = \{[S \rightarrow B\bullet]\}$$

$$
\begin{aligned}
q_5 &= \delta(q_0, A) &= \{[S \to A \bullet b\, B],\\
&&[B \to A \bullet]\}
\end{aligned}
\qquad
\begin{aligned}
q_8 &= \delta(q_5, b) &= \{[S \to A\, b \bullet B],\\
&&[B \to \bullet A],\\
&&[A \to \bullet a],\\
&&[A \to \bullet b\, B]\}
\end{aligned}
$$

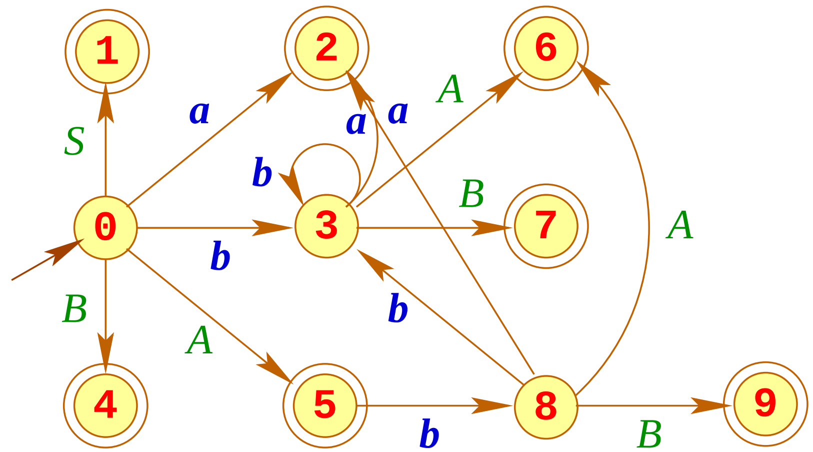
$$
q_6 = \delta(q_3, A) = \{[B \to A \bullet]\}
$$

$$
q_7 = \delta(q_3, B) = \{[A \to b\, B \bullet]\} \qquad q_9 = \delta(q_8, B) = \{[S \to A\, b\, B \bullet]\}
$$

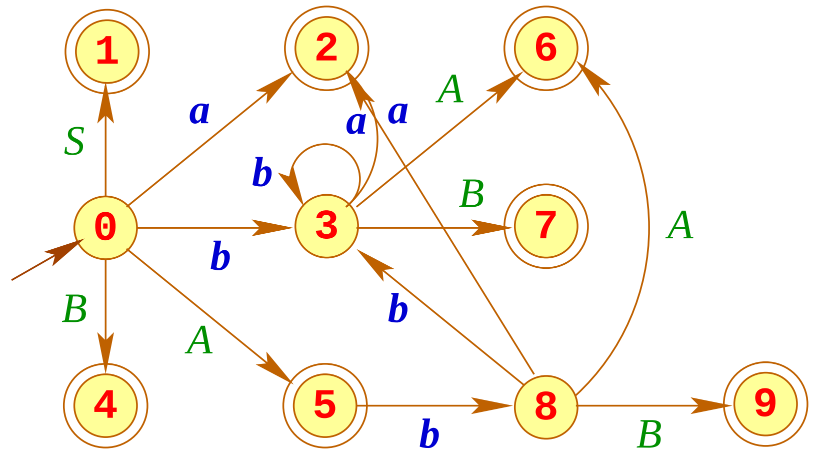
Shift-Reduce-Konflikt:

$$
\begin{aligned}
q_5 = \{&[S \to A \bullet b\, B],\\
&[B \to A \bullet]\}
\end{aligned}
$$

Dabei ist: $\mathsf{Follow}_1(B) \cap \{b\} \odot \{\ldots\} = \{\epsilon, b\} \cap \{b\} \neq \emptyset$



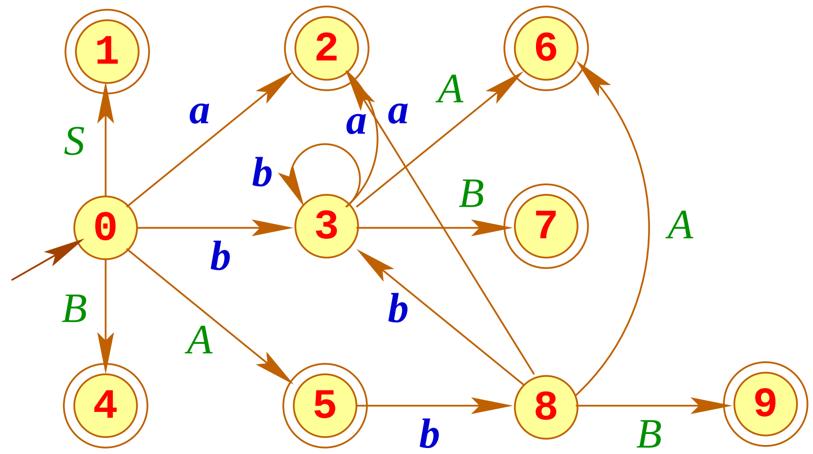
1
2
6
S
a
a
a
A
b
B
0
3
7
A
b
b
B
A
4
5
8
9
b
B



Ausschnitt des Ungleichungssystems:

$$\Lambda_1(q_5, [B \to A\bullet]) \quad \supseteq \quad \Lambda_1(q_0, [B \to \bullet\, A])$$

$$\begin{array}{lll}
\Lambda_1(q_0, [B \to \bullet\, A]) & \supseteq & \Lambda_1(q_0, [S \to \bullet\, B]) \\
\Lambda_1(q_0, [S \to \bullet\, B]) & \supseteq & \Lambda_1(q_0, [S' \to \bullet\, S]) \\
\Lambda_1(q_0, [S' \to \bullet\, S]) & \supseteq & \{\epsilon\}
\end{array}$$



Ausschnitt des Ungleichungssystems:

$$\Lambda_1(q_1, [B \to A\bullet]) \quad \supseteq \quad \Lambda_1(q_0, [B \to \bullet\, A])$$

$$\begin{aligned}
\Lambda_1(q_0, [B \to \bullet\, A]) \quad &\supseteq \quad \Lambda_1(q_0, [S \to \bullet\, B]) \\
\Lambda_1(q_0, [S \to \bullet\, B]) \quad &\supseteq \quad \Lambda_1(q_0, [S' \to \bullet\, S]) \\
\Lambda_1(q_0, [S' \to \bullet\, S]) \quad &\supseteq \quad \{\epsilon\}
\end{aligned}$$

Folglich: $\quad \Lambda_1(q_5, [B \to A\bullet]) \; = \; \{\epsilon\}$

Diskussion:

- Das Beispiel ist folglich nicht $SLR(1)$, aber $LALR(1)$:-)
- Das Beispiel ist nicht so an den Haaren herbei gezogen, wie es scheint ...
- Umbenennung: $A \Rightarrow L \quad B \Rightarrow R \quad a \Rightarrow \mathsf{id} \quad b \Rightarrow * \,/ =$ liefert:

$$
\begin{array}{lll}
S & \rightarrow & L = R \mid R \\
L & \rightarrow & \mathsf{id} \mid * R \\
R & \rightarrow & L
\end{array}
$$

... d.h. ein Fragment der Grammatik für C-Ausdrücke ;-)

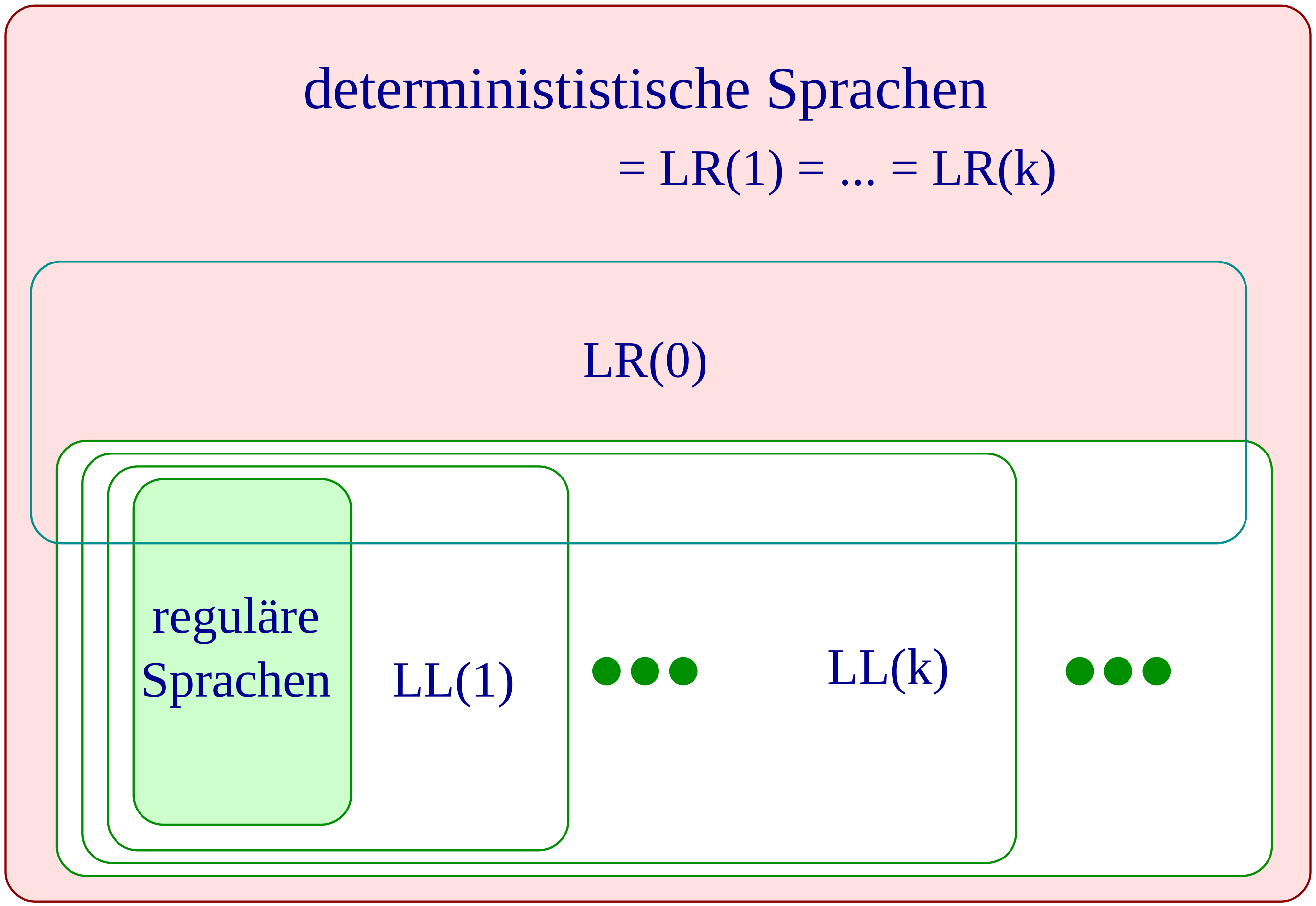
Für $k = 1$ lassen sich die Mengen $\Lambda_k(q, [A \to \alpha \bullet \beta])$ wieder effizient berechnen :-)

Das verbesserte Ungleichungsssystem:

$$
\begin{array}{lcll}
\Lambda_1(q_0, [S' \to \bullet\, S]) & \supseteq & \{\epsilon\} & \\
\Lambda_1(q, [A \to \alpha\, X \bullet \beta]) & \supseteq & \Lambda_1(p, [A \to \alpha \bullet\, X\, \beta]) & \text{falls} \quad \delta(p, X) = q \\
\Lambda_1(q, [A \to \bullet\, \gamma]) & \supseteq & F_\epsilon(X_j) & \text{falls} \quad [B \to \alpha \bullet A\, X_1 \ldots X_m] \in q \\
 & & & \text{und} \quad \mathsf{empty}(X_1) \wedge \ldots \wedge \mathsf{empty}(X_{j-1}) \\
\Lambda_1(q, [A \to \bullet\, \gamma]) & \supseteq & \Lambda_1(q, [B \to \alpha \bullet A\, X_1 \ldots X_m]) & \text{falls} \quad [B \to \alpha \bullet A\, X_1 \ldots X_m] \in q \\
 & & & \text{und} \quad \mathsf{empty}(X_1) \wedge \ldots \wedge \mathsf{empty}(X_m)
\end{array}
$$

$\Longrightarrow$ ein reines Vereinigungsproblem :-))

Übersicht über die Sprachklassen:



Diskussion:

- Alle kontextfreien Sprachen, die sich mit einem deterministischen Kellerautomaten parsen lassen, können durch eine LR(1)-Grammatik beschrieben werden.
- Durch LR(0)-Grammatiken lassen sich alle präfixfreien deterministisch kontextfreien Sprachen beschreiben :-)
- Die Sprachklassen zu LL(k)-Grammatiken bilden dagegen eine Hierarchie innerhalb der deterministisch kontextfreien Sprachen.
- Da zu jeder LL(k)-Grammatik eine äquivalente starke LL(k)-Grammatik konstruiert werden kann, sinde letztere nicht in der Übersicht vermerkt.

3 Semantische Analyse

- Lexikalisch und syntaktisch korrekte Programme können trotzdem fehlerhaft sein ;-(
- Einige von diesen Fehlern werden bereits durch die Sprachdefinition ausgeschlossen und müssen vom Compiler überprüft werden :-)
- Weitere Analysen sind erforderlich, um:

  $\longrightarrow$ Bezeichner eindeutig zu machen;

  $\longrightarrow$ die Typen von Variablen zu ermitteln;

  $\longrightarrow$ Möglichkeiten zur Programm-Optimierung zu finden.

3.1 Symbol-Tabellen

Beispiel:

```
void foo() {
    int A;
    void fee() {
        double A;
        A = 0.5;
        write(A);
    }
    A = 2;
    fee();
    write(A);
}
```

Diskussion:

- Innerhalb des Rumpfs von fee wird die Definition von A durch die lokale Definition verdeckt :-)
- Für die Code-Erzeugung benötigen wir für jede Benutzung eines Bezeichners die zugehörige Definitionsstelle.
- Statische Bindung bedeutet, dass dass die Definition eines Namens A an allen Programmpunkten innerhalb ihres gesamten Blocks gültig ist.
- Sichtbar ist sie aber nur außerhalb derjenigen Teilbereiche, in an denen eine weitere Definition von A gültig ist :-)

... im Beispiel:

```
void foo() {
    int A;
    void fee() {
        double A;
        A = 0.5;
        write(A);
    }
    A = 2;
    fee();
    write(A);
}
```

Kompliziertere Regeln der Sichtbarkeit gibt es in objektorientierten Programmiersprachen wie Java ...

Beispiel:

```
public class Foo {
        protected int x = 17;
        protected int y = 5;
        private int z = 42;
        public int b() { return 1; }
    }
class Fee extends Foo {
        protected double y = .5;
        public int b(int a) { return a; }
    }
```

Diskussion:

- **private** Members sind nur innerhalb der aktuellen Klasse gültig :-)
- **protected** Members sind innerhalb der Klasse, in den Unterklassen sowie innerhalb des gesamten package gültig :-)
- Methoden b gleichen Namens sind stets verschieden, wenn ihre Argument-Typen verschieden sind !!!
- Bei Aufrufen einer Methode wird dynamisch entschieden, welche Definition gemeint ist ...