Offenbar gilt:

- Gilt $F^k \underline{\perp} = F^{k+1} \underline{\perp}$, ist eine Lösung gefunden :-)
- $\underline{\perp}, F \underline{\perp}, F^2 \underline{\perp}, \ldots$ bilden eine aufsteigende Kette :

$$\underline{\perp} \quad \sqsubseteq \quad F \underline{\perp} \quad \sqsubseteq \quad F^2 \underline{\perp} \quad \sqsubseteq \quad \ldots$$

- Sind alle aufsteigenden Ketten endlich, gibt es k immer.

Offenbar gilt:

- Gilt $F^k \underline{\perp} = F^{k+1} \underline{\perp}$, ist eine Lösung gefunden :-)
- $\underline{\perp}, F \underline{\perp}, F^2 \underline{\perp}, \ldots$ bilden eine aufsteigende Kette :

$$\underline{\perp} \quad \sqsubseteq \quad F \underline{\perp} \quad \sqsubseteq \quad F^2 \underline{\perp} \quad \sqsubseteq \quad \ldots$$

- Sind alle aufsteigenden Ketten endlich, gibt es k immer.

Die zweite Aussage folgt mit vollständiger Induktion:

Offenbar gilt:

- Gilt $F^k \underline{\perp} = F^{k+1} \underline{\perp}$, ist eine Lösung gefunden :-)
- $\underline{\perp}, F \underline{\perp}, F^2 \underline{\perp}, \ldots$ bilden eine aufsteigende Kette :

$$\underline{\perp} \quad \sqsubseteq \quad F \underline{\perp} \quad \sqsubseteq \quad F^2 \underline{\perp} \quad \sqsubseteq \quad \ldots$$

- Sind alle aufsteigenden Ketten endlich, gibt es k immer.

Die zweite Aussage folgt mit vollständiger Induktion:

Anfang: $F^0 \underline{\perp} = \underline{\perp} \sqsubseteq F^1 \underline{\perp}$:-)

Offenbar gilt:

- Gilt $F^k \underline{\perp} = F^{k+1} \underline{\perp}$, ist eine Lösung gefunden :-)
- $\underline{\perp}, F \underline{\perp}, F^2 \underline{\perp}, \ldots$ bilden eine aufsteigende Kette :

$$\underline{\perp} \quad \sqsubseteq \quad F \underline{\perp} \quad \sqsubseteq \quad F^2 \underline{\perp} \quad \sqsubseteq \quad \ldots$$

- Sind alle aufsteigenden Ketten endlich, gibt es k immer.

Die zweite Aussage folgt mit vollständiger Induktion:

Anfang: $F^0 \underline{\perp} = \underline{\perp} \sqsubseteq F^1 \underline{\perp}$:-)

Schluss: Gelte bereits $F^{i-1} \underline{\perp} \sqsubseteq F^i \underline{\perp}$. Dann

$$F^i \underline{\perp} = F\,(F^{i-1} \underline{\perp}) \sqsubseteq F\,(F^i \underline{\perp}) = F^{i+1} \underline{\perp}$$

da F monoton ist :-)

Fazit:

Wenn $\mathbb{D}$ endlich ist, finden wir mit Sicherheit eine Lösung :-)

Fragen:

Fazit:

Wenn $\mathbb{D}$ endlich ist, finden wir mit Sicherheit eine Lösung :-)

Fragen:

1. Gibt es eine kleinste Lösung ?

Fazit:

Wenn $\mathbb{D}$ endlich ist, finden wir mit Sicherheit eine Lösung :-)

Fragen:

1. Gibt es eine kleinste Lösung ?
2. Wenn ja: findet Iteration die kleinste Lösung ??

Fazit:

Wenn $\mathbb{D}$ endlich ist, finden wir mit Sicherheit eine Lösung :-)

Fragen:

1. Gibt es eine kleinste Lösung ?
2. Wenn ja: findet Iteration die kleinste Lösung ??
3. Was, wenn $\mathbb{D}$ nicht endlich ist ???

Satz — Kleene

In einer vollständigen Halbordnung $\mathbb{D}$ hat jede stetige Funktion $f : \mathbb{D} \to \mathbb{D}$ einen kleinsten Fixpunkt d_0.

Dieser ist gegeben durch $d_0 = \bigsqcup_{k \geq 0} f^k \bot$.

Satz — Kleene

In einer vollständigen Halbordnung $\mathbb{D}$ hat jede stetige Funktion $f : \mathbb{D} \to \mathbb{D}$ einen kleinsten Fixpunkt d_0.

Dieser ist gegeben durch $d_0 = \bigsqcup_{k \geq 0} f^k \bot$.

Bemerkung:

- Eine Funktion f heißt stetig, falls für jede aufsteigende Kette $d_0 \sqsubseteq \ldots \sqsubseteq d_m \sqsubseteq \ldots$ gilt: $f\,(\bigsqcup_{m \geq 0} d_m) = \bigsqcup_{m \geq 0} (f\, d_m)$.
- Werden alle aufsteigenden Ketten irgendwann stabil, ist jede monotone Funktion automatisch stetig :-)

Satz — Kleene

In einer vollständigen Halbordnung $\mathbb{D}$ hat jede stetige Funktion $f : \mathbb{D} \to \mathbb{D}$ einen kleinsten Fixpunkt d_0.

Dieser ist gegeben durch $d_0 = \bigsqcup_{k \geq 0} f^k \bot$.

Bemerkung:

- Eine Funktion f heißt stetig, falls für jede aufsteigende Kette $d_0 \sqsubseteq \ldots \sqsubseteq d_m \sqsubseteq \ldots$ gilt: $f\,(\bigsqcup_{m \geq 0} d_m) = \bigsqcup_{m \geq 0} (f\, d_m)$.
- Werden alle aufsteigenden Ketten irgendwann stabil, ist jede monotone Funktion automatisch stetig :-)
- Eine Halbordnung heißt vollständig (CPO), falls alle aufsteigenden Ketten kleinste obere Schranken haben :-)
- Jeder vollständige Verband ist auch eine vollständige Halbordnung :-)

Beweis:

(1) $f\,d_0 = d_0$:

$$\begin{aligned} f\,d_0 &= f\left(\bigsqcup_{m\geq 0}(f^m\bot)\right) \\ &= \bigsqcup_{m\geq 0}(f^{m+1}\bot) \qquad \text{wegen Stetigkeit} \quad \text{:-)} \\ &= \bot \sqcup \left(\bigsqcup_{m\geq 0}(f^{m+1}\bot)\right) \\ &= \bigsqcup_{m\geq 0}(f^m\bot) \\ &= d_0 \end{aligned}$$

(2) d_0 ist kleinster Fixpunkt:

Sei $f\,d_1 = d_1$ weiterer Fixpunkt. Wir zeigen: $\forall\, m \geq 0 :\ f^m\bot \sqsubseteq d_1$.

$m = 0$:

$$\bot \sqsubseteq d_1 \qquad \text{nach Definition}$$

$m > 0$: Gelte

$$\begin{aligned} f^{m-1}\bot &\sqsubseteq d_1 \qquad \text{Dann folgt:} \\ f^m\bot &= f\,(f^{m-1}\bot) \\ &\sqsubseteq f\,d_1 \qquad \text{wegen Monotonie} \quad \text{:-)} \\ &= d_1 \end{aligned}$$

Bemerkung:

- Jede stetige Funktion ist auch monoton :-)
- Betrachte die Menge der Postfixpunkte:

$$P = \{x \in \mathbb{D} \mid x \sqsupseteq f\,x\}$$

Der kleinste Fixpunkt d_0 ist in P und untere Schranke :-)

$\Longrightarrow$ d_0 ist der kleinste Wert x mit $x \sqsupseteq f\,x$

Bemerkung:

- Jede stetige Funktion ist auch monoton :-)
- Betrachte die Menge der Postfixpunkte:

$$P = \{x \in \mathbb{D} \mid x \sqsupseteq f\,x\}$$

Der kleinste Fixpunkt d_0 ist in P und untere Schranke :-)

$\Longrightarrow$ d_0 ist der kleinste Wert x mit $x \sqsupseteq f\,x$

Anwendung:

Sei $$x_i \sqsupseteq f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \qquad (*)$$

ein Ungleichungssystem, wobei alle $f_i : \mathbb{D}^n \to \mathbb{D}$ monoton sind.

Bemerkung:

- Jede stetige Funktion ist auch monoton :-)
- Betrachte die Menge der Postfixpunkte:

$$P = \{x \in \mathbb{D} \mid x \sqsupseteq f\,x\}$$

Der kleinste Fixpunkt d_0 ist in P und untere Schranke :-)

$\Longrightarrow$ d_0 ist der kleinste Wert x mit $x \sqsupseteq f\,x$

Anwendung:

Sei $$x_i \sqsupseteq f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \qquad (*)$$

ein Ungleichungssystem, wobei alle $f_i : \mathbb{D}^n \to \mathbb{D}$ monoton sind.

$\Longrightarrow$ kleinste Lösung von $(*)$ $=$ kleinster Fixpunkt von F :-)

Der Kleenesche Fixpunkt-Satz liefert uns nicht nur die Existenz einer kleinsten Lösung sondern auch eine Charakterisierung :-)

Satz

Die Mengen $\mathsf{First}_k(\{w \in T^* \mid A \to^* w\})$, $A \in N$, sind die kleinste Lösung des Ungleichungssystems:

$$\mathsf{First}_k(A) \quad \supseteq \quad \mathsf{First}_k(X_1) \odot \ldots \odot \mathsf{First}_k(X_m) \ , \quad A \to X_1 \ldots X_m \quad \in \quad P$$

Der Kleenesche Fixpunkt-Satz liefert uns nicht nur die Existenz einer kleinsten Lösung sondern auch eine Charakterisierung :-)

Satz

Die Mengen $\mathsf{First}_k(\{w \in T^* \mid A \to^* w\})$, $A \in N$, sind die kleinste Lösung des Ungleichungssystems:

$$\mathsf{First}_k(A) \quad \supseteq \quad \mathsf{First}_k(X_1) \odot \ldots \odot \mathsf{First}_k(X_m) \ , \quad A \to X_1 \ldots X_m \quad \in \quad P$$

Beweis-Idee:

Sei $\mathsf{F}^{(m)}(A)$ die m-te Approximation an den Fixpunkt.

(1) Falls $A \to^m u$, dann $\mathsf{First}_k(u) \subseteq \mathsf{F}^{(m)}(A)$.

(2) Falls $w \in \mathsf{F}^{(m)}(A)$, dann $A \to^* u$ für $u \in T^*$ mit $\mathsf{First}_k(u) = \{w\}$:-)

Fazit:

Wir können First_k durch Fixpunkt-Iteration berechnen, d.h. durch wiederholtes Einsetzen :-)

Fazit:

Wir können First_k durch Fixpunkt-Iteration berechnen, d.h. durch wiederholtes Einsetzen :-)

Achtung:

Naive Fixpunkt-Iteration ist ziemlich ineffizient :-(

Fazit:

Wir können First_k durch Fixpunkt-Iteration berechnen, d.h. durch wiederholtes Einsetzen :-)

Achtung: Naive Fixpunkt-Iteration ist ziemlich ineffizient :-(

Idee: Round Robin Iteration

Benutze bei der Iteration nicht die Werte der letzten Iteration, sondern die jeweils aktuellen :-)

Unser Mini-Beispiel: $\mathbb{D} = 2^{\{a,b,c\}}, \quad \sqsubseteq \; = \; \subseteq$

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

Die Round-Robin-Iteration:

	1	2	3
x_1	$\{a\}$	$\{a, c\}$	dito
x_2	$\emptyset$	$\{a\}$	
x_3	$\{a, c\}$	$\{a, c\}$	

Der Code für Round Robin Iteration sieht in Java so aus:

```
for (i = 1; i ≤ n; i++) x_i = ⊥;
do {
    finished = true;
    for (i = 1; i ≤ n; i++) {
        new = f_i(x_1, ..., x_n);
        if (!(x_i ⊒ new)) {
            finished = false;
            x_i = x_i ⊔ new;
        }
    }
} while (!finished);
```

Zur Korrektheit:

Sei $y_i^{(d)}$ die i-te Komponente von $F^d \underline{\perp}$.

Sei $x_i^{(d)}$ der Wert von x_i nach der i-ten RR-Iteration.

Zur Korrektheit:

Sei $y_i^{(d)}$ die i-te Komponente von $F^d\ \underline{\perp}$.

Sei $x_i^{(d)}$ der Wert von x_i nach der i-ten RR-Iteration.

Man zeigt:

(1) $y_i^{(d)} \sqsubseteq x_i^{(d)}$:-)

Zur Korrektheit:

Sei $y_i^{(d)}$ die i-te Komponente von $F^d \underline{\perp}$.

Sei $x_i^{(d)}$ der Wert von x_i nach der i-ten RR-Iteration.

Man zeigt:

(1) $y_i^{(d)} \sqsubseteq x_i^{(d)}$:-)

(2) $x_i^{(d)} \sqsubseteq z_i$ für jede Lösung $(z_1, \ldots, z_n)$:-)

Zur Korrektheit:

Sei $y_i^{(d)}$ die i-te Komponente von $F^d\ \underline{\perp}$.

Sei $x_i^{(d)}$ der Wert von x_i nach der i-ten RR-Iteration.

Man zeigt:

(1) $y_i^{(d)} \sqsubseteq x_i^{(d)}$:-)

(2) $x_i^{(d)} \sqsubseteq z_i$ für jede Lösung $(z_1, \ldots, z_n)$:-)

(3) Terminiert RR-Iteration nach d Runden, ist
$(x_1^{(d)}, \ldots, x_n^{(d)})$ eine Lösung :-))

Unsere Anwendung:

$$
\begin{array}{lcll}
\mathsf{First}_2(E) & \supseteq & \mathsf{First}_2(E) \odot \{+\} \odot \mathsf{First}_2(T) & \cup \quad \mathsf{First}_2(T) \\
\mathsf{First}_2(T) & \supseteq & \mathsf{First}_2(T) \odot \{*\} \odot \mathsf{First}_2(F) & \cup \quad \mathsf{First}_2(F) \\
\mathsf{First}_2(F) & \supseteq & \{(\} \odot \mathsf{First}_2(E) \odot \{)\} & \cup \quad \{\mathsf{name}, \mathsf{int}\}
\end{array}
$$

Die RR-Iteration:

First_2	1	2	3
F	name, int	(name, (int	((
T	name, int	(name, (int, name $*$, int $*$	((
E	name, int	(name, (int, name $*$, int $*$, name $+$, int $+$	((

Der Einfachkeit halber haben wir in jeder Iteration nur die neuen Elemente vermerkt :-)

Diskussion:

- Die Länge h der längsten echt aufsteigenden Kette nennen wir auch Höhe von $\mathbb{D}$ ...
- Im Falle von First_k ist die Höhe des Verbands exponentiell in k :-(
- Die Anzahl der Runden von RR-Iteration ist beschränkt durch $\mathcal{O}(n \cdot h)$ (n die Anzahl der Variablen)
- Die praktische Effizienz von RR-Iteration hängt allerdings auch von der Anordnung der Variablen ab :-)
- Anstelle von RR-Iteration gibt es auch schnellere Fixpunkt-Verfahren, die aber im schlimmsten Fall immer noch exponentiell sind :-((

$\Longrightarrow$ Man beschränkt sich i.a. auf kleine k !!!

2.4 Topdown Parsing

Idee:

- Benutze den Item-Kellerautomaten.
- Benutze die nächsten k Zeichen, um die Regeln für die Expansionen zu bestimmen ;-)
- Eine Grammatik heißt $LL(k)$, falls dies immer eindeutig möglich ist.

2.4 Topdown Parsing

Idee:

- Benutze den Item-Kellerautomaten.
- Benutze die nächsten k Zeichen, um die Regeln für die Expansionen zu bestimmen ;-)
- Eine Grammatik heißt $LL(k)$, falls dies immer eindeutig möglich ist.

Wir definieren:

Eine reduzierte Grammatik heißt dann $LL(k)$, falls für je zwei verschiedene Regeln $A \to \alpha$, $A \to \alpha' \in P$ und jede Ableitung $S \to_L^* u\, A\, \beta$ mit $u \in T^*$ gilt:

$$\mathsf{First}_k(\alpha\,\beta) \cap \mathsf{First}_k(\alpha'\,\beta) = \emptyset$$

Beispiel 1:

$$
\begin{array}{lcl}
S & \rightarrow & \mathsf{if}\ (\ E\ )\ S\ \mathsf{else}\ S \quad | \\
 & & \mathsf{while}\ (\ E\ )\ S \quad | \\
 & & E\ ; \\
E & \rightarrow & \mathsf{id}
\end{array}
$$

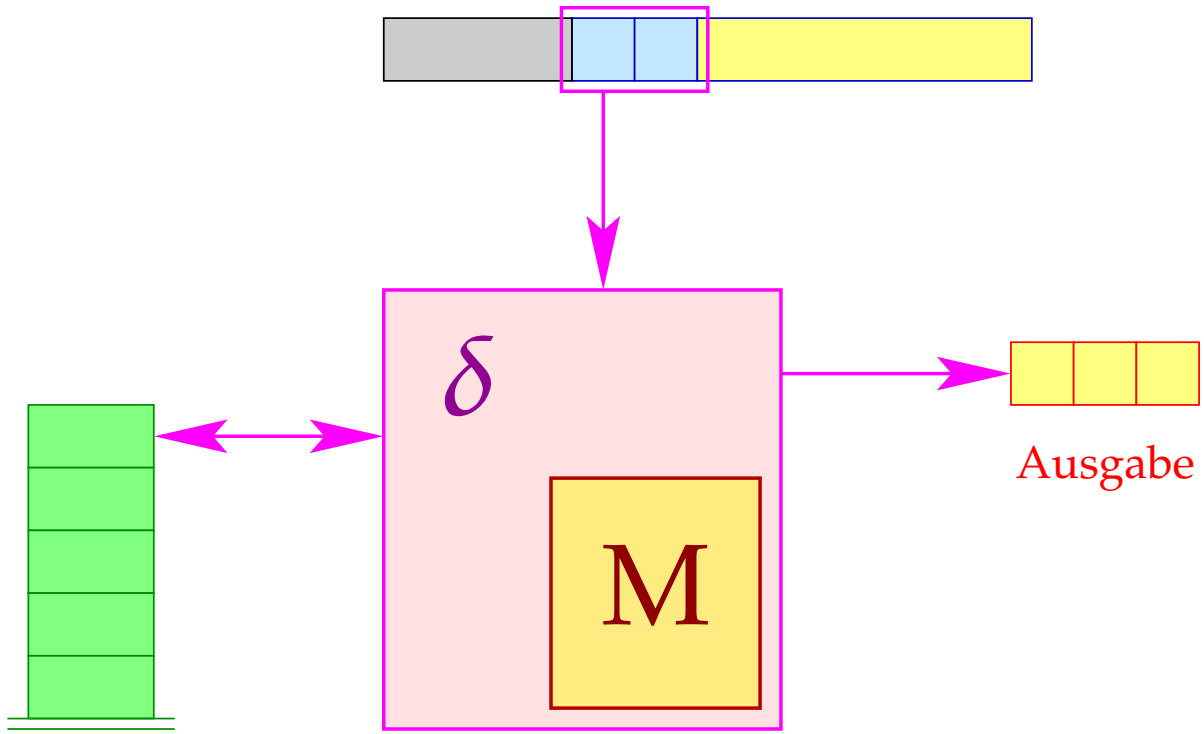
ist $LL(1)$, da $\quad \mathsf{First}_k(E) = \{\mathsf{id}\}$:-)

Beispiel 2:

$$
\begin{array}{lcl}
S & \rightarrow & \text{if } (\ E \ ) \ S \text{ else } S \quad | \\
 & & \text{if } (\ E \ ) \ S \quad | \\
 & & \text{while } (\ E \ ) \ S \quad | \\
 & & E \ ; \\
E & \rightarrow & \text{id}
\end{array}
$$

... ist nicht $LL(k)$ für jedes $k > 0$.

Struktur des $LL(k)$-Parsers:



- Der Parser sieht ein Fenster der Länge k der Eingabe;
- er realisiert im Wesentlichen den Item-Kellerautomaten;
- die Tabelle $M[q, w]$ enthält die jeweils zuwählende Regel :-)

... im Beispiel:

$$
\begin{array}{lcl}
S & \rightarrow & \text{if}\ (\ E\ )\ S\ \text{else}\ S\ ^{0} \quad | \\
 & & \text{while}\ (\ E\ )\ S\ ^{1} \quad | \\
 & & E\ ;\ ^{2} \\
E & \rightarrow & \text{id}\ ^{0}
\end{array}
$$

Zustände: Items

Tabelle:

	if	while	id
$[\ldots \rightarrow \ldots \bullet S \ldots]$	0	1	2
$[\ldots \rightarrow \ldots \bullet E \ldots]$	–	–	0

Im Allgemeinen ...

- ist die Menge der möglichen nächsten k Zeichen gegeben durch:

$$\mathsf{First}_k(\alpha\,\beta) \;=\; \mathsf{First}_k(\alpha) \odot \mathsf{First}_k(\beta)$$

  wobei:

  (1) α die rechte Seite der passenden Regel;

  (2) β ein möglicher rechter Kontext von A ist :-)

- $\mathsf{First}_k(\beta)$ müssen wir dynamisch akkumulieren.

$\Longrightarrow$ Wir erweitern Items um Vorausschau-Mengen ...

Ein erweitertes Item ist ein Paar: $[A \to \alpha \bullet \gamma\,,\ L]$ $(A \to \alpha\,\gamma \in P,\ L \subseteq T^{\leq k})$

Die Menge L benutzen wir, um $\mathsf{First}_k(\beta)$ für den rechten Kontext β von A zu repräsentieren :-)

Ein erweitertes Item ist ein Paar: $[A \to \alpha \bullet \gamma\,,\ L]$ $(A \to \alpha\,\gamma \in P,\ L \subseteq T^{\leq k})$

Die Menge L benutzen wir, um $\mathsf{First}_k(\beta)$ für den rechten Kontext β von A zu repräsentieren :-)

Konstruktion:

Zustände: erweiterte Items

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

Endzustand: $[S' \to S\,\bullet,\ \{\epsilon\}]$

Übergänge:

Ein erweitertes Item ist ein Paar: $[A \to \alpha \bullet \gamma\,,\ L]$ $(A \to \alpha\,\gamma \in P,\ L \subseteq T^{\leq k})$

Die Menge L benutzen wir, um $\mathsf{First}_k(\beta)$ für den rechten Kontext β von A zu repräsentieren :-)

Konstruktion:

Zustände: erweiterte Items

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

Endzustand: $[S' \to S\,\bullet,\ \{\epsilon\}]$

Übergänge:

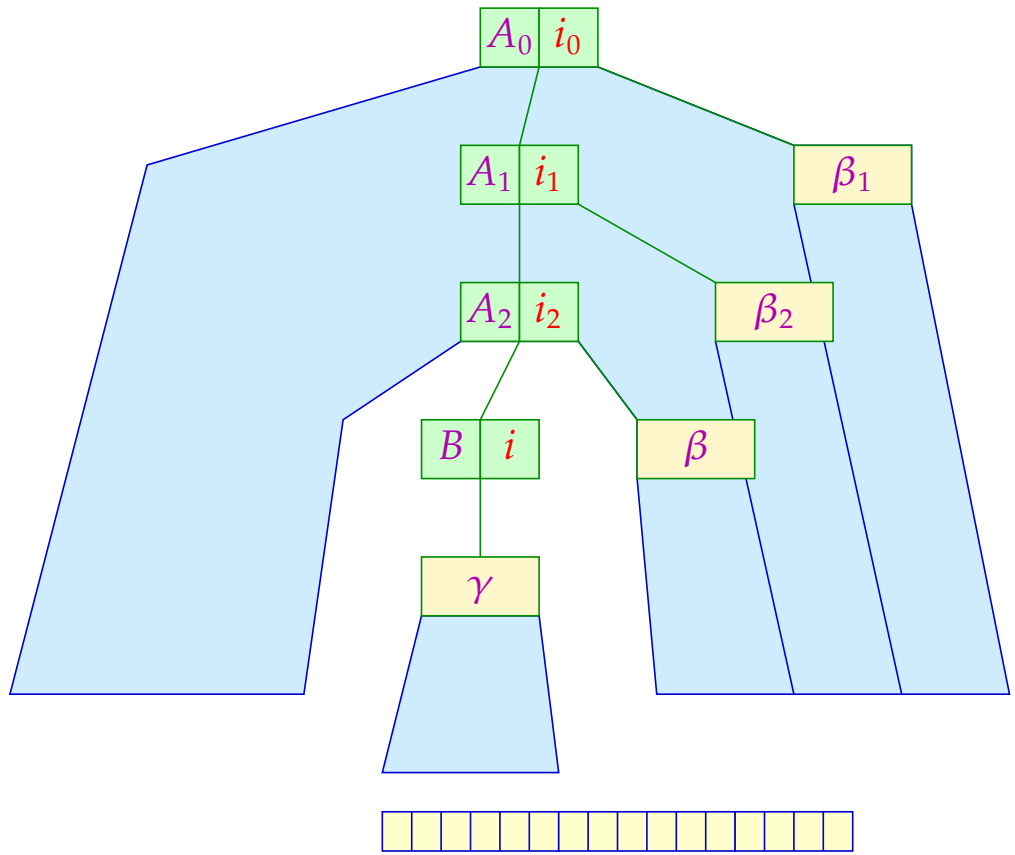
Expansionen: $([A \to \alpha \bullet B\,\beta\,,\ L], \epsilon, [A \to \alpha \bullet B\,\beta\,,\ L]\,[B \to \bullet\,\gamma\,,\ \boxed{\mathsf{First}_k(\beta) \odot L}])$

für $A \to \alpha\,B\,\beta,\ B \to \gamma \in P$

Shifts: $([A \to \alpha \bullet a\,\beta\,,\ L], a, [A \to \alpha\,a \bullet \beta\,,\ L])$ für $A \to \alpha\,a\,\beta \in P$

Reduce: $([A \to \alpha \bullet B\,\beta\,,\ L]\,[B \to \gamma\bullet\,,\ L'], \epsilon, [A \to \alpha\,B \bullet \beta\,,\ L])$ für

$A \to \alpha\,B\,\beta,\ B \to \gamma \in P$



Die Vorausschau-Tabelle:

Wir setzen $M[\,[A \to \alpha \bullet B\,\beta\,,\ L],\ w] \ = \ i$ genau dann wenn (B, i) die Regel $B \to \gamma$ ist und: $w \in \mathsf{First}_k(\gamma) \odot \mathsf{First}_k(\beta) \odot L$

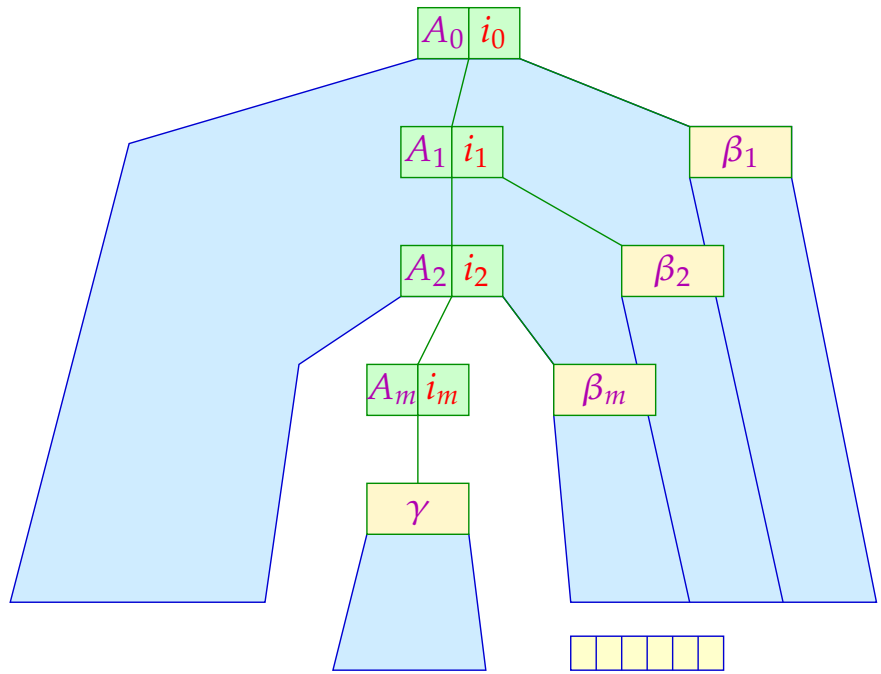
$$([A_0 \to \bullet\, \alpha_1\, A_1\, \beta_1,\ L_1],\ uv) \quad \vdash^* \quad ([A_0 \to \alpha_1 \bullet A_1\, \beta_1,\ L_1] \ldots [A_{m-1} \to \alpha_m \bullet A_m\, \beta_m,\ L_m],\ v)$$

$$\vdash^* \quad ([A_0 \to \alpha_1\, A_1\, \beta_1 \bullet,\ L_1],\ \epsilon) \quad \ldots \quad \text{gilt genau dann wenn:}$$

(1) $\alpha_1 \ldots \alpha_m \to^* u$

(2) $A_m\, \beta_m \ldots \beta_1 \to^* v$

(3) $L_m = \mathsf{First}_k(\beta_{m-1}) \odot \ldots \odot \mathsf{First}_k(\beta_1) \odot L_1$



Satz

Die reduzierte kontextfreie Grammatik G ist $LL(k)$ genau dann wenn die k-Vorausschau-Tabelle für alle benötigten erweiterten Items wohl-definiert ist.

Diskussion:

- Der erweiterte Item-Kellerautomat zusammen mit einer k-Vorausschau-Tabelle erlaubt die deterministische Rekonstruktion einer Links-Ableitung :-)
- Die Anzahl der Vorausschau-Mengen L kann sehr groß sein :-(
- ...

Beispiel: $S \to \epsilon \quad | \quad a\,S\,b$

Die Übergänge des erweiterten Item-Kellerautomat $(k = 1)$:

0	$[S' \to \bullet\, S, \{\epsilon\}]$	ϵ	$[S' \to \bullet\, S, \{\epsilon\}]\,[S \to \bullet, \{\epsilon\}]$
1	$[S' \to \bullet\, S, \{\epsilon\}]$	ϵ	$[S' \to \bullet\, S, \{\epsilon\}]\,[S \to \bullet\, a\,S\,b, \{\epsilon\}]$
2	$[S \to \bullet\, a\,S\,b, \{\epsilon\}]$	a	$[S \to a \bullet S\,b, \{\epsilon\}]$
	$[S \to \bullet\, a\,S\,b, \{b\}]$	a	$[S \to a \bullet S\,b, \{b\}]$
3	$[S \to a \bullet S\,b, \{\epsilon\}]$	ϵ	$[S \to a \bullet S\,b, \{\epsilon\}]\,[S \to \bullet, \{b\}]$
	$[S \to a \bullet S\,b, \{b\}]$	ϵ	$[S \to a \bullet S\,b, \{b\}]\,[S \to \bullet, \{b\}]$
4	$[S \to a \bullet S\,b.\{\epsilon\}]$	ϵ	$[S \to a \bullet S\,b.\{\epsilon\}]\,[S \to \bullet\, a\,S\,b, \{b\}]$
	$[S \to a \bullet S\,b.\{b\}]$	ϵ	$[S \to a \bullet S\,b.\{b\}]\,[S \to \bullet\, a\,S\,b, \{b\}]$
5	$[S \to a \bullet S\,b, \{\epsilon\}]\,[S \to \bullet, \{b\}]$	ϵ	$[S \to a\,S \bullet b, \{\epsilon\}]$
	$[S \to a \bullet S\,b, \{b\}]\,[S \to \bullet, \{b\}]$	ϵ	$[S \to a\,S \bullet b, \{b\}]$
	...		...

	...		...
6	$[S \rightarrow a \bullet S\, b, \{\epsilon\}]\, [S \rightarrow a\, S\, b \bullet, \{b\}]$	ϵ	$[S \rightarrow a\, S \bullet b, \{\epsilon\}]$
	$[S \rightarrow a \bullet S\, b, \{b\}]\, [S \rightarrow a\, S\, b \bullet, \{b\}]$	ϵ	$[S \rightarrow a\, S \bullet b, \{b\}]$
7	$[S \rightarrow a\, S \bullet b, \{\epsilon\}]$	b	$[S \rightarrow a\, S\, b \bullet, \{\epsilon\}]$
	$[S \rightarrow a\, S \bullet b, \{b\}]$	b	$[S \rightarrow a\, S\, b \bullet, \{b\}]$
8	$[S' \rightarrow \bullet S, \{\epsilon\}]\, [S \rightarrow \bullet, \{\epsilon\}]$	ϵ	$[S' \rightarrow S \bullet, \{\epsilon\}]$
9	$[S' \rightarrow \bullet S, \{\epsilon\}]\, [S \rightarrow a\, S\, b \bullet, \{\epsilon\}]$	ϵ	$[S' \rightarrow S \bullet, \{\epsilon\}]$

Die Vorausschau-Tabelle:

	ϵ	a	b
$[S' \rightarrow \bullet S, \{\epsilon\}]$	0	1	–
$[S \rightarrow a \bullet S\, b, \{\epsilon\}]$	–	1	0
$[S \rightarrow a \bullet S\, b, \{b\}]$	–	1	0