

Offenbar gilt:

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

$$\perp \sqsubseteq F\perp \sqsubseteq F^2\perp \sqsubseteq \dots$$

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

Offenbar gilt:

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

$$\perp \sqsubseteq F\perp \sqsubseteq F^2\perp \sqsubseteq \dots$$

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

Die zweite Aussage folgt mit **vollständiger Induktion**:

Offenbar gilt:

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

$$\perp \sqsubseteq F\perp \sqsubseteq F^2\perp \sqsubseteq \dots$$

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

Die zweite Aussage folgt mit **vollständiger Induktion**:

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

Offenbar gilt:

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

$$\perp \sqsubseteq F \perp \sqsubseteq F^2 \perp \sqsubseteq \dots$$

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

Die zweite Aussage folgt mit **vollständiger Induktion**:

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

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

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

da F monoton ist :-)

Fazit:

Wenn  endlich ist, finden wir mit Sicherheit eine Lösung :-)

Fragen:

Fazit:

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

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 $f : \mathbb{D} \rightarrow \mathbb{D}$ einen **kleinsten Fixpunkt** d_0 .

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

Satz

Kleene

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

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

Bemerkung:

- Eine Funktion f heißt **stetig**, falls für jede aufsteigende Kette $d_0 \sqsubseteq \dots \sqsubseteq d_m \sqsubseteq \dots$ 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 $f : \mathbb{D} \rightarrow \mathbb{D}$ einen **kleinsten Fixpunkt** d_0 .

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

Bemerkung:

- Eine Funktion f heißt **stetig**, falls für jede aufsteigende Kette $d_0 \sqsubseteq \dots \sqsubseteq d_m \sqsubseteq \dots$ 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:

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

(2) d_0 ist **kleinster** Fixpunkt:

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

$m = 0$: $\perp \sqsubseteq d_1$ nach Definition

$m > 0$: Gelte $f^{m-1} \perp \sqsubseteq d_1$ Dann folgt:

$$\begin{aligned}f^m \perp &= f(f^{m-1} \perp) \\ &\sqsubseteq f d_1 \quad \text{wegen Monotonie :-)} \\ &= d_1\end{aligned}$$

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** :-)

$\implies d_0$ 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** :-)

$\implies d_0$ ist der kleinste Wert x mit $x \sqsupseteq f x$

Anwendung:

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

ein **Ungleichungssystem**, wobei alle $f_i : \mathbb{D}^n \rightarrow \mathbb{D}$ 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** :-)

====> d_0 ist der kleinste Wert x mit $x \sqsupseteq f x$

Anwendung:

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

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

====> 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 $\text{First}_k(\{w \in T^* \mid A \rightarrow^* w\})$, $A \in N$, sind die kleinste Lösung des Ungleichungssystems:

$$\text{First}_k(A) \supseteq \text{First}_k(X_1) \odot \dots \odot \text{First}_k(X_m), \quad A \rightarrow X_1 \dots X_m \in P$$

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

Satz

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

$$\text{First}_k(A) \supseteq \text{First}_k(X_1) \odot \dots \odot \text{First}_k(X_m), \quad A \rightarrow X_1 \dots X_m \in P$$

Beweis-Idee:

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

- (1) Falls $A \rightarrow^m u$, dann $\text{First}_k(u) \subseteq F^{(m)}(A)$.
- (2) Falls $w \in F^{(m)}(A)$, dann $A \rightarrow^* u$ für $u \in T^*$ mit $\text{First}_k(u) = \{w\}$:-)

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: $\mathbb{D} = 2^{\{\textcolor{red}{a}, \textcolor{red}{b}, \textcolor{red}{c}\}}, \sqsubseteq = \subseteq$

$$\begin{aligned}x_1 &\supseteq \{\textcolor{red}{a}\} \cup x_3 \\x_2 &\supseteq x_3 \cap \{\textcolor{red}{a}, \textcolor{red}{b}\} \\x_3 &\supseteq x_1 \cup \{\textcolor{red}{c}\}\end{aligned}$$

Die Round-Robin-Iteration:

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

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

```
for ( $i = 1; i \leq n; i++$ )  $x_i = \perp$ ;
do {
     $finished = \text{true}$ ;
    for ( $i = 1; i \leq n; i++$ ) {
         $new = f_i(x_1, \dots, x_n)$ ;
        if ( $!(x_i \sqsupseteq new)$ ) {
             $finished = \text{false}$ ;
             $x_i = x_i \sqcup new$ ;
        }
    }
} while ( $!finished$ );
```

Zur Korrektheit:

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

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 $F^d \perp$.

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

Man zeigt:

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

Zur Korrektheit:

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

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

Man zeigt:

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

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

Zur Korrektheit:

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

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

Man zeigt:

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

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

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

Unsere Anwendung:

$$\begin{aligned}\text{First}_2(E) &\supseteq \text{First}_2(E) \odot \{+\} \odot \text{First}_2(T) \cup \text{First}_2(T) \\ \text{First}_2(T) &\supseteq \text{First}_2(T) \odot \{* \} \odot \text{First}_2(F) \cup \text{First}_2(F) \\ \text{First}_2(F) &\supseteq \{() \odot \text{First}_2(E) \odot ()\} \cup \{\text{name, int}\}\end{aligned}$$

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 Einfachheit 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 :-((

====> 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 $A \rightarrow \alpha, A \rightarrow \alpha' \in P$ und jede Ableitung $S \rightarrow_L^* u A \beta$ mit $u \in T^*$ gilt:

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

Beispiel 1:

$S \rightarrow \text{if (} E \text{) } S \text{ else } S \quad |$
 $\text{while (} E \text{) } S \quad |$
 $E ;$
 $E \rightarrow \text{id}$

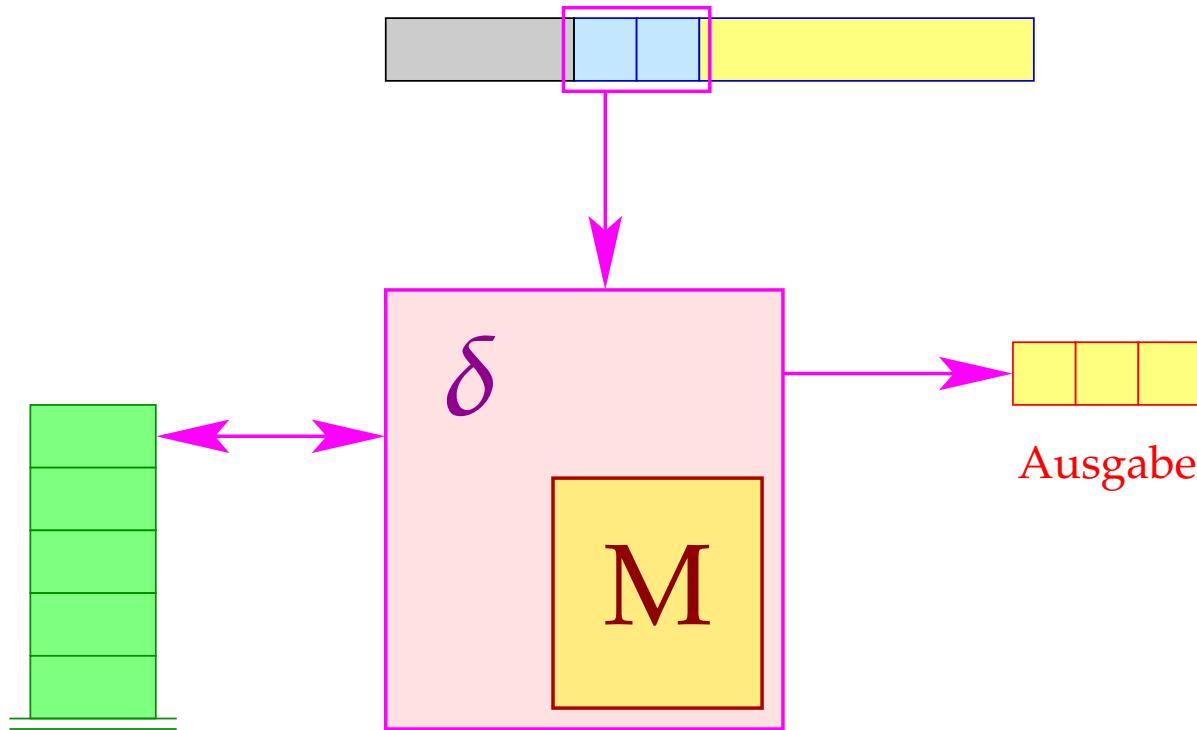
ist $LL(1)$, da $\text{First}_k(E) = \{\text{id}\} \quad :-)$

Beispiel 2:

$S \rightarrow \text{if (} E \text{) } S \text{ else } S \quad |$
 $\quad \quad \quad \text{if (} E \text{) } S \quad |$
 $\quad \quad \quad \text{while (} E \text{) } S \quad |$
 $\quad \quad \quad E ;$
 $E \rightarrow \text{id}$

... 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:

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

Zustände: Items

Tabelle:

	if	while	id
[... \rightarrow ... \bullet S ...]	0	1	2
[... \rightarrow ... \bullet E ...]	—	—	0

Im Allgemeinen ...

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

$$\text{First}_k(\alpha\beta) = \text{First}_k(\alpha) \odot \text{First}_k(\beta)$$

wobei:

- (1) α die rechte Seite der passenden Regel;
 - (2) β ein möglicher rechter Kontext von A ist $\text{:-})$
- $\text{First}_k(\beta)$ müssen wir **dynamisch** akkumulieren.

====> Wir erweitern Items um Vorausschau-Mengen ...

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

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

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

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

Konstruktion:

Zustände: erweiterte Items

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

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

Übergänge:

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

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

Konstruktion:

Zustände: erweiterte Items

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

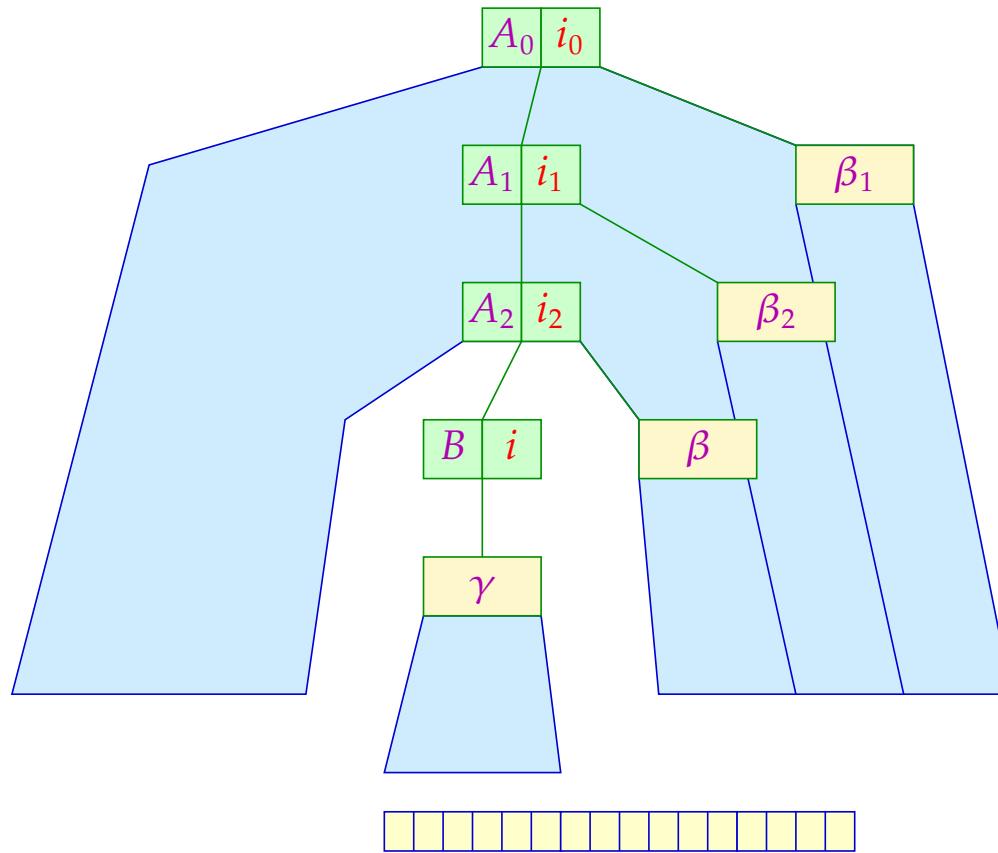
Endzustand: $[S' \rightarrow S \bullet, \{\epsilon\}]$

Übergänge:

Expansionen: $([A \rightarrow \alpha \bullet B \beta, L], \epsilon, [A \rightarrow \alpha \bullet B \beta, L] [B \rightarrow \bullet \gamma, \text{First}_k(\beta) \odot L])$
für $A \rightarrow \alpha B \beta, B \rightarrow \gamma \in P$

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

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

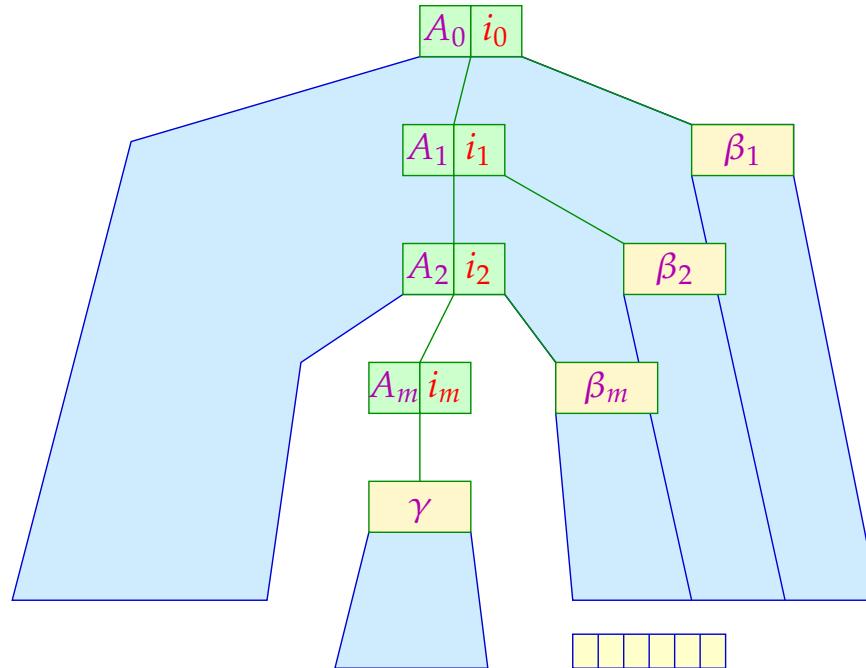


Die Vorausschau-Tabelle:

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

$([A_0 \rightarrow \bullet \alpha_1 A_1 \beta_1, L_1], uv) \vdash^* ([A_0 \rightarrow \alpha_1 \bullet A_1 \beta_1, L_1] \dots [A_{m-1} \rightarrow \alpha_m \bullet A_m \beta_m, L_m], v)$
 $\vdash^* ([A_0 \rightarrow \alpha_1 A_1 \beta_1 \bullet, L_1], e) \dots$ gilt genau dann wenn:

- (1) $\alpha_1 \dots \alpha_m \rightarrow^* u$
- (2) $A_m \beta_m \dots \beta_1 \rightarrow^* v$
- (3) $L_m = \text{First}_k(\beta_{m-1}) \odot \dots \odot \text{First}_k(\beta_1) \odot L_1$



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 \vdash)
- Die Anzahl der Vorausschau-Mengen L kann sehr groß sein \vdash (
- ...

Beispiel:

$$S \rightarrow \epsilon \quad | \quad a S b$$

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

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

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