

## Allgemein:

$$\text{Follow}_k(S) \supseteq \{\epsilon\}$$

$$\text{Follow}_k(B) \supseteq \text{First}_k(X_1) \odot \dots \odot \text{First}_k(X_m) \odot \text{Follow}_k(A)$$

für  $A \xrightarrow{\alpha} B X_1 \dots X_m \in P$

## Diskussion:

- Man überzeugt sich, dass die **kleinste** Lösung dieses Ungleichungssystems tatsächlich die Mengen  $\text{Follow}_k(B)$  liefert :-)
- Die Größe der auftretenden Mengen steigt mit  $k$  rapide :-(
- In praktischen Systemen wird darum meist nur der Fall  $k = 1$  implementiert ...

## 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:

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

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

## Idee:

- Behandle  $\epsilon$  separat!

Sei  $\text{empty}(X) = \text{true}$  gdw.  $X \xrightarrow{*} \epsilon$ .

- Definiere die  $\epsilon$ -freien  $\text{First}_1$ -Mengen

$$F_\epsilon(a) = \{a\} \quad \text{für } a \in T$$

$$F_\epsilon(A) = \text{First}_1(A) \setminus \{\epsilon\} \quad \text{für } A \in N$$

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

$$F_\epsilon(A) \supseteq F_\epsilon(X_j) \quad \text{falls } A \xrightarrow{*} X_1 \dots X_m \in P,$$

$$\text{empty}(X_1) \wedge \dots \wedge \text{empty}(X_{j-1})$$

... im Beispiel:

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

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

Deshalb erhalten wir:

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

Entsprechend konstruieren wir zur Berechnung von  $\text{Follow}_1$  :

$$\text{Follow}_1(S) \supseteq \{\epsilon\}$$

$$\text{Follow}_1(B) \supseteq F_\epsilon(X_j) \quad \text{falls} \quad A \rightarrow \alpha B X_1 \dots X_m \in P, \\ \text{empty}(X_1) \wedge \dots \wedge \text{empty}(X_{j-1})$$

$$\text{Follow}_1(B) \supseteq \text{Follow}_1(A) \quad \text{falls} \quad A \rightarrow \alpha B X_1 \dots X_m \in P, \\ \text{empty}(X_1) \wedge \dots \wedge \text{empty}(X_m)$$

Entsprechend konstruieren wir zur Berechnung von  $\text{Follow}_1$  :

$$\text{Follow}_1(S) \supseteq \{\epsilon\}$$

$$\text{Follow}_1(B) \supseteq F_\epsilon(X_j) \quad \text{falls} \quad A \rightarrow \alpha B X_1 \dots X_m \in P, \\ \text{empty}(X_1) \wedge \dots \wedge \text{empty}(X_{j-1})$$

$$\text{Follow}_1(B) \supseteq \text{Follow}_1(A) \quad \text{falls} \quad A \rightarrow \alpha B X_1 \dots X_m \in P, \\ \text{empty}(X_1) \wedge \dots \wedge \text{empty}(X_m)$$

... im Beispiel:

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

Entsprechend konstruieren wir zur Berechnung von  $\text{Follow}_1$ :

$$\text{Follow}_1(S) \supseteq \{\epsilon\}$$

$$\text{Follow}_1(B) \supseteq F_\epsilon(X_j) \quad \text{falls } A \rightarrow \alpha B X_1 \dots X_m \in P, \\ \text{empty}(X_1) \wedge \dots \wedge \text{empty}(X_{j-1})$$

$$\text{Follow}_1(B) \supseteq \text{Follow}_1(A) \quad \text{falls } A \rightarrow \alpha B X_1 \dots X_m \in P, \\ \text{empty}(X_1) \wedge \dots \wedge \text{empty}(X_m)$$

... im Beispiel:

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

... erhalten wir:

$$\text{Follow}_1(S') \supseteq \{\epsilon\}$$

$$\text{Follow}_1(E) \supseteq \{+, )\}$$

$$\text{Follow}_1(T) \supseteq \text{Follow}_1(E)$$

$$\text{Follow}_1(E) \supseteq \text{Follow}_1(S')$$

$$\text{Follow}_1(T) \supseteq \{\ast\}$$

$$\text{Follow}_1(F) \supseteq \text{Follow}_1(T)$$

## Diskussion:

- Diese Ungleichungssysteme bestehen aus Ungleichungen der Form:

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

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

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

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

$$x_0 \supseteq \{a\}$$

$$x_1 \supseteq \{b\}$$

$$x_1 \supseteq x_0$$

$$x_1 \supseteq x_3$$

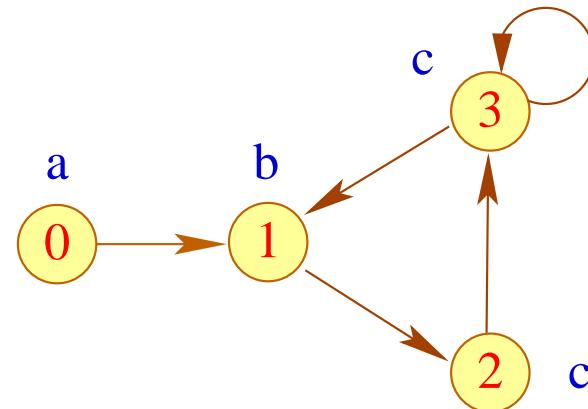
$$x_2 \supseteq \{c\}$$

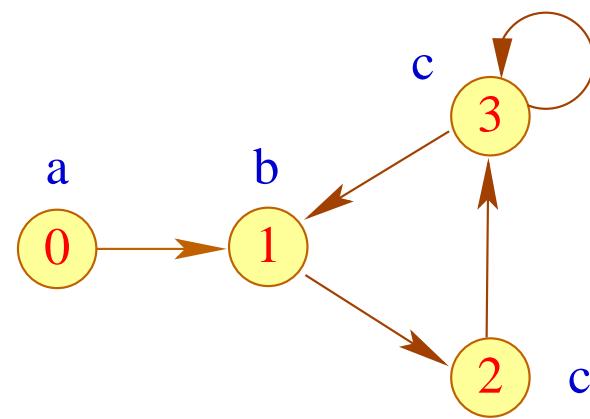
$$x_2 \supseteq x_1$$

$$x_3 \supseteq \{c\}$$

$$x_3 \supseteq x_2$$

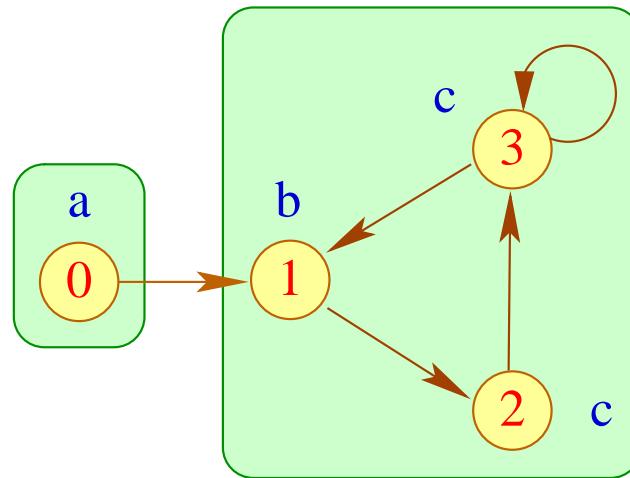
$$x_3 \supseteq x_3$$





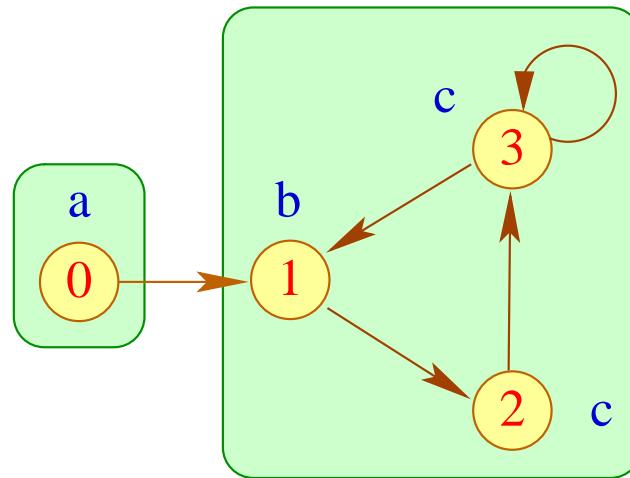
Vorgehen:

- Konstruiere den Variablen-Abhangigkeitsgraph zum Ungleichungssystem.



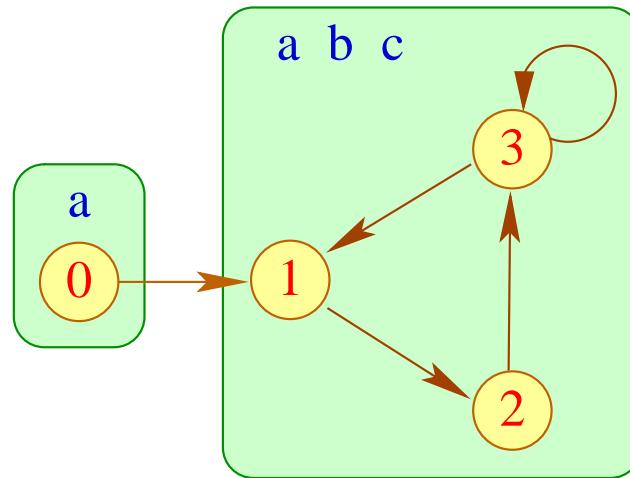
## Vorgehen:

- Konstruiere den **Variablen-Abhangigkeitsgraph** zum Ungleichungssystem.
- Innerhalb einer **starken Zusammenhangskomponente** haben alle Variablen den gleichen Wert :-)



## Vorgehen:

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

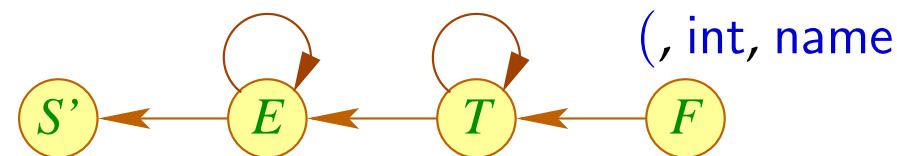


## Vorgehen:

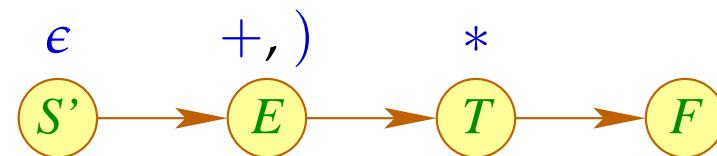
- Konstruiere den **Variablen-Abhangigkeitsgraph** zum Ungleichungssystem.
- Innerhalb einer **starken Zusammenhangskomponente** haben alle Variablen den gleichen Wert :-)
- Hat eine SZK keine eingehenden Kanten, erhalt man ihren Wert, indem man die kleinste obere Schranke aller Werte in der SZK berechnet :-)
- Gibt es eingehende Kanten, muss man zusatzlich die Werte an deren Startknoten hinzufugen :-)

... für unsere Beispiel-Grammatik:

First<sub>1</sub> :



Follow<sub>1</sub> :



## 2.6 Bottom-up Analyse

Achtung:

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

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

## 2.6 Bottom-up Analyse

Achtung:

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

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

Beispiel:

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

... 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:  $A \rightarrow A\beta \in P$

$A$  erreichbar  $\implies S \rightarrow_L^* u A \gamma \rightarrow_L^* u A \beta^n \gamma$  für jedes  $n \geq 0$ .

$A$  produktiv  $\implies \exists A \rightarrow \alpha : \alpha \neq A\beta$ .

## Satz

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

**Beweis:** Vereinfachung:  $A \rightarrow A\beta \in P$

$A$  erreichbar  $\implies S \xrightarrow{L}^* u A \gamma \xrightarrow{L}^* u A \beta^n \gamma$  für jedes  $n \geq 0$ .

$A$  produktiv  $\implies \exists A \rightarrow \alpha : \alpha \neq A\beta$ .

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

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

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

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

## Satz

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

**Beweis:** Vereinfachung:  $A \rightarrow A\beta \in P$

$A$  erreichbar  $\implies S \xrightarrow{L}^* u A \gamma \xrightarrow{L}^* u A \beta^n \gamma$  für jedes  $n \geq 0$ .

$A$  produktiv  $\implies \exists A \rightarrow \alpha : \alpha \neq A\beta$ .

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

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

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

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

**Fall 1:**  $\beta \xrightarrow{*} \epsilon$  — Widerspruch !!!

**Fall 2:**  $\beta \xrightarrow{*} w \neq \epsilon \implies \text{First}_k(\alpha\beta^k\gamma) \cap \text{First}_k(\alpha\beta^{k+1}\gamma) \neq \emptyset$  :-)

## Bottom-up Parsing:

Wir rekonstruieren reverse Rechtsableitungen  $\text{:-})$

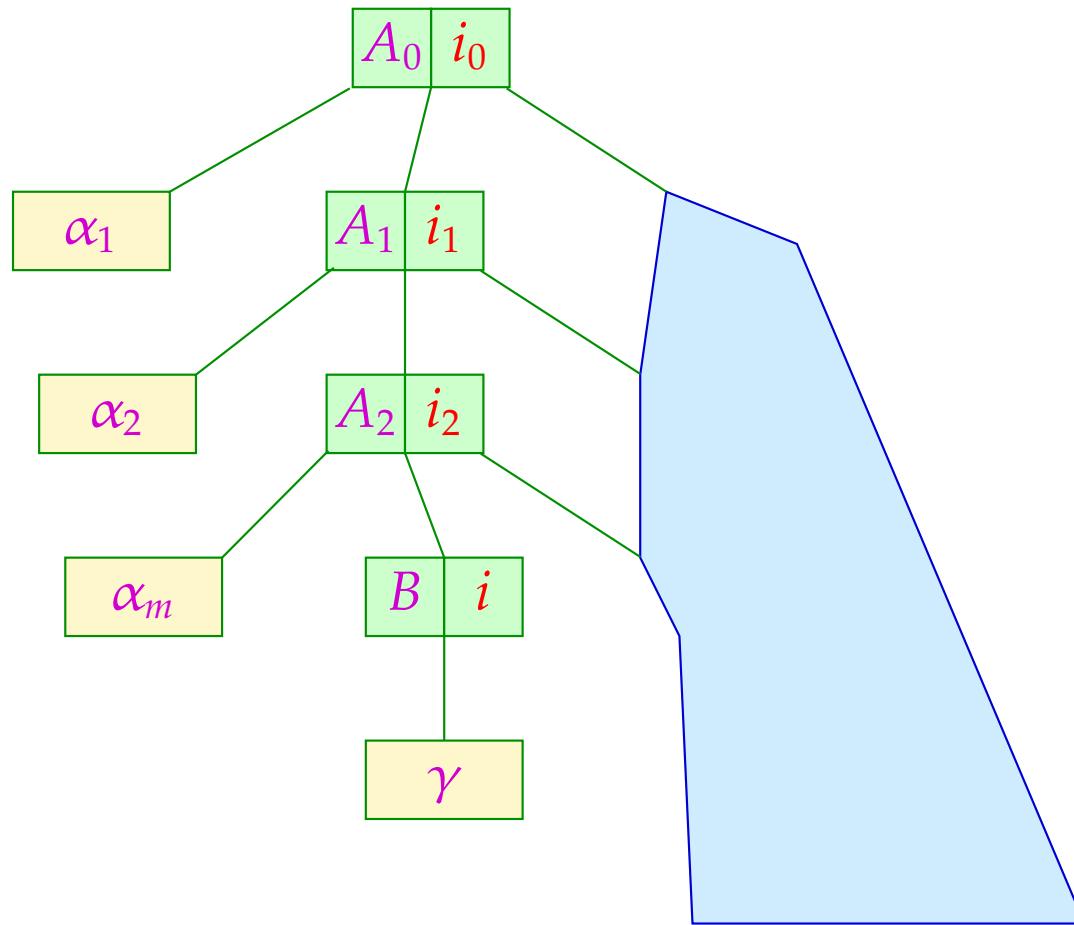
Dazu versuchen wir, für den Shift-Reduce-Parser  $M_G^{(1)}$  die Reduktionsstellen zu identifizieren ...

Betrachte eine Berechnung dieses Kellerautomaten:

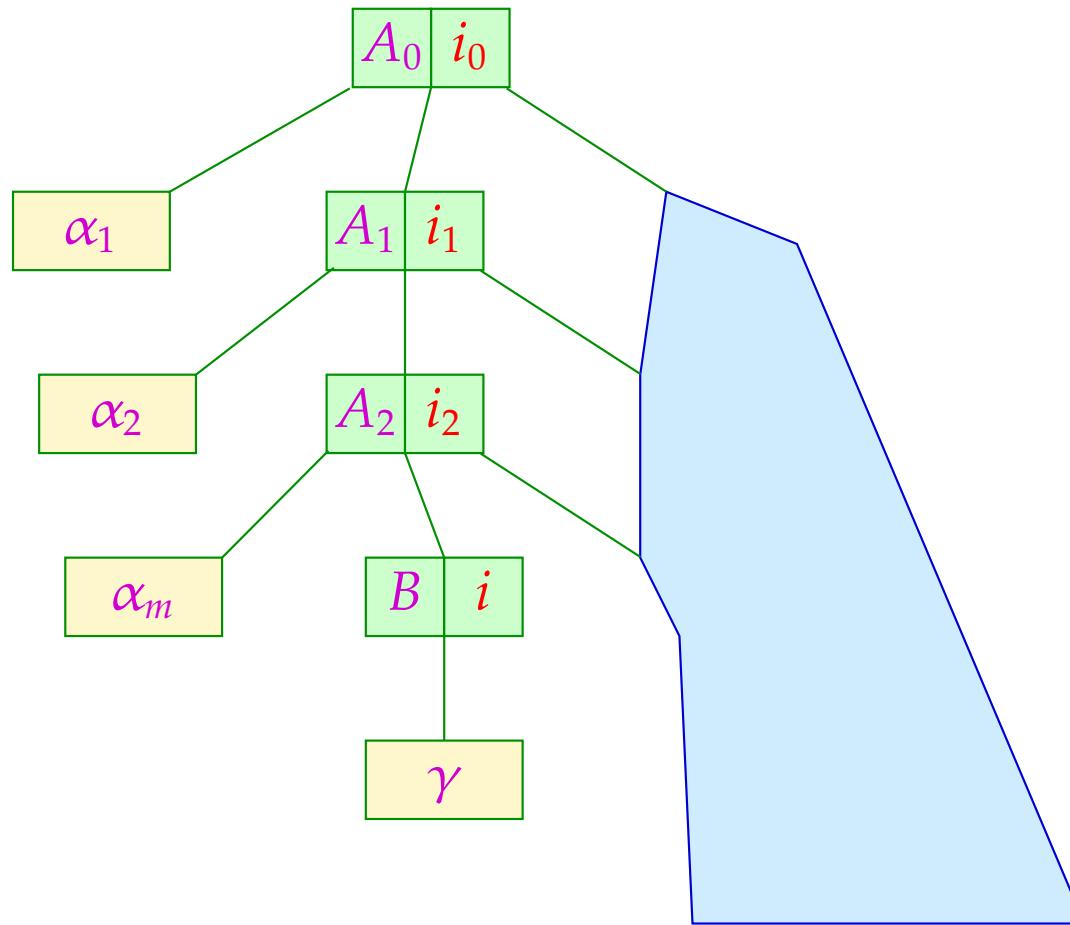
$$(q_0 \alpha \gamma, v) \vdash (q_0 \alpha B, v) \vdash^* (q_0 S, \epsilon)$$

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

Dann ist  $\alpha \gamma$  zuverlässig für  $[B \rightarrow \gamma \bullet]$  gdw.  $S \xrightarrow{R}^* \alpha B v \text{ :-})$



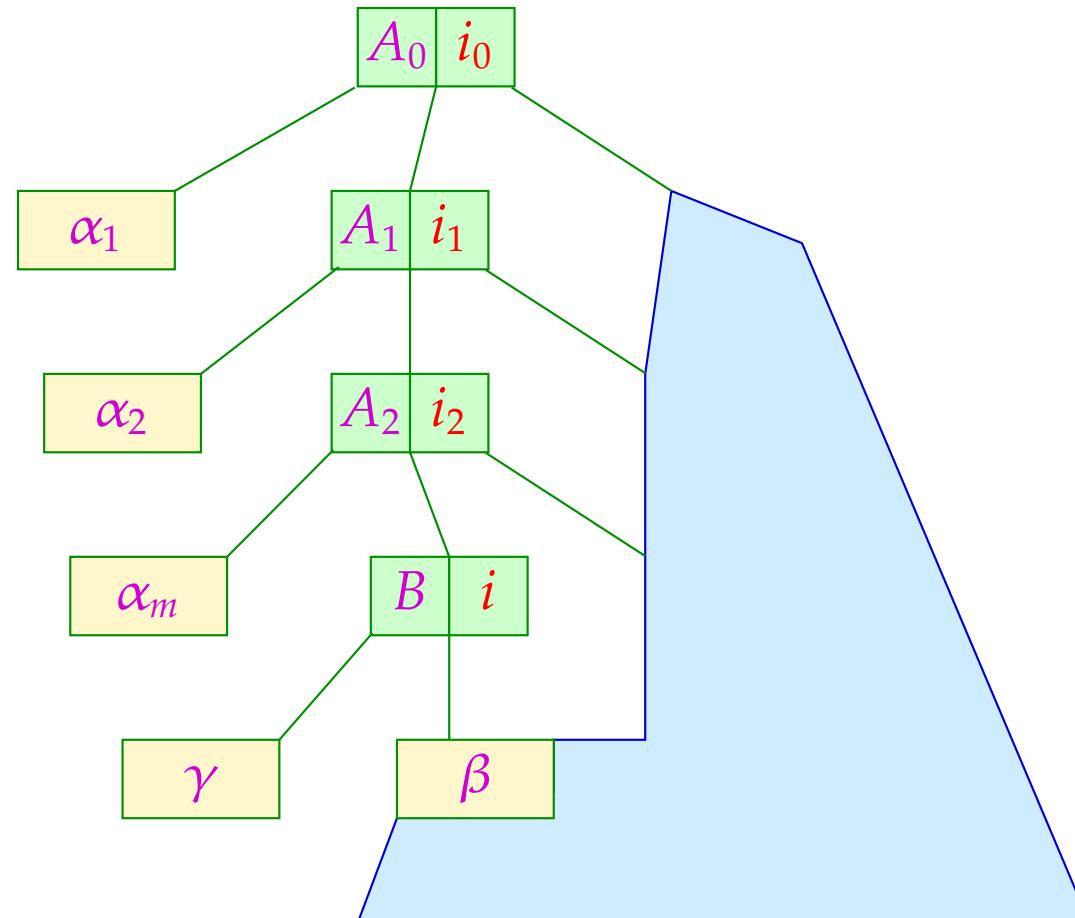
... wobei  $\alpha = \alpha_1 \dots \alpha_m \text{ :-)}$



... wobei  $\alpha = \alpha_1 \dots \alpha_m \quad :-)$

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

Das Item  $[B \rightarrow \gamma \bullet \beta]$  heißt **gültig** für  $\alpha'$  gdw.  $S \xrightarrow{R}^* \alpha B \nu$  mit  $\alpha' = \alpha \gamma$ :



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

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

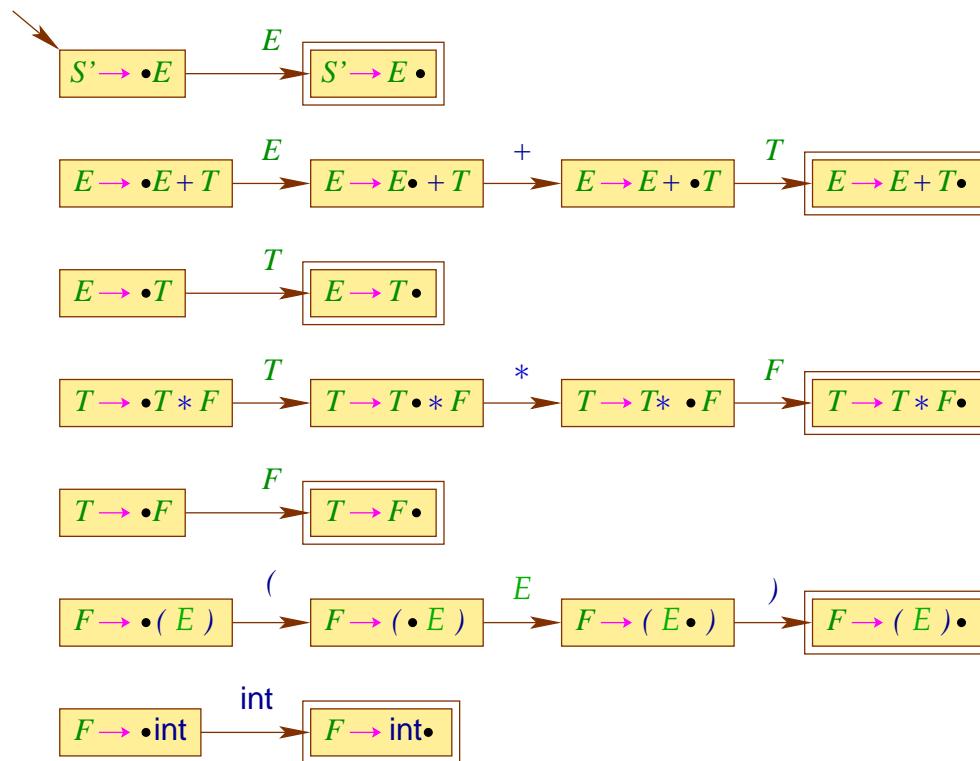
Den Automaten  $c(G)$  nennen wir **charakteristischen Automaten** für  $G$ .

## Beispiel:

$$E \rightarrow E + T \quad | \quad T$$

$$T \rightarrow T * F \quad | \quad F$$

$$F \rightarrow (E) \quad | \quad \text{int}$$



## Beispiel:

$E$	$\rightarrow$	$E + T$	$ $	$T$
$T$	$\rightarrow$	$T * F$	$ $	$F$
$F$	$\rightarrow$	$( E )$	$ $	int

