

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



Vorlesung Compilerbau: Syntaktische Analyse – Parsing VI LR/LR(k)-Parsing

Vorlesung des BA-Studiums
Prof. Johann Christoph Freytag, Ph.D.
Institut für Informatik, Humboldt-Universität zu Berlin
SoSe 2018

© Prof. J.C. Freytag, Ph.D.

9.1

LR(k)-Grammatiken

Informal:

- eine Grammatik G ist vom Typ $LR(k)$, falls ein von links nach rechts arbeitender Shift-Reduce-Parser in der Lage ist, für eine gegebene Rechtsableitung

$$S \Rightarrow \gamma_0 \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \dots \Rightarrow \gamma_{n-1} \Rightarrow \gamma_n = w$$

und für jede rechte Satzform γ_i in der Ableitung

- den Handle in dieser rechten Satzform zu finden und
- die Reduktionsregel zu bestimmen,

wobei er maximal k Symbole über das rechte Ende der Handle für γ_i hinausschauen darf, um seine Shift-Reduce-Entscheidungen zu treffen

© Prof. J.C. Freytag, Ph.D.

9.2

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

LR(k)-Grammatiken (2)

Annahme:

sei $G' = (V_N', V_T, P', S')$ folgende Erweiterung bzgl. einer kontextfreie Grammatik $G = (V_N, V_T, P, S)$: S' ist neues Startsymbol mit $S' \Rightarrow S$

Definition: G heißt LR(k)-Grammatik, wenn aus

- $S' \Rightarrow^* \alpha A w \Rightarrow \alpha \beta w$ und
- $S' \Rightarrow^* \gamma B x \Rightarrow \alpha \beta y$ und
- $\text{FIRST}_k(w) = \text{FIRST}_k(y)$

folgt, dass

- $\alpha = \gamma$
- $A = B$
- $x = y$.

© Prof. J.C. Freytag, Ph.D.

9.3

LR(k)-Grammatiken (3)

Definition: Grammatik G heißt LR(k)-Grammatik, wenn aus

- $S' \Rightarrow^* \alpha A w \Rightarrow \alpha \beta w$ und
- $S' \Rightarrow^* \gamma B x \Rightarrow \alpha \beta y$ und
- $\text{FIRST}_k(w) = \text{FIRST}_k(y)$

folgt, dass

- $\alpha = \gamma$, $A = B$ und $x = y$.

Plausibilität:

- Wir nehmen an, dass die Satzformen $\alpha \beta w$ und $\alpha \beta y$ mit gemeinsamen Präfix $\alpha \beta$ und gleichem »k-Lookahead« an Symbolen konstruiert sind,

d.h. $\text{FIRST}_k(y) = \text{FIRST}_k(w)$,
so dass $\alpha \beta w$ zu $\alpha A w$ bzw. $\alpha \beta y$ zu $\alpha B y$ reduziert werden kann

- aus dem gemeinsamen Präfix folgt, dass $\alpha \beta y$ auch zu $\alpha A y$ mit dem gleichen Ergebnis reduziert werden kann
- deshalb gilt $\alpha A y = \gamma B x$

© Prof. J.C. Freytag, Ph.D.

9.4

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



Besonderheiten von LR(k)-Grammatiken

- LR-Grammatiken sind die *allgemeinsten* Grammatiken, deren Sätze von einem **deterministischen, Bottom-Up-Parser** erkannt werden können
- LR-Parser erkennen Fehler, sobald dies in einem Von-Links-nach-Rechts-Lesen der Eingabe möglich ist
- LR(k)-Grammatiken beschreiben eine echte Obermenge der Sprachen, die durch LL(k)-Parser erkannt werden:
 - **LL(k)**: erkennt die Anwendung der Regel $A \rightarrow \beta$, wenn sie die ersten k Symbole gesehen hat, die von β abgeleitet werden können
 - **LR(k)**: erkennt die Handle β , nachdem sie alles gesehen hat, was aus β abgeleitet werden kann plus (maximal) k Lookahead-Symbole
- Für jede LR(k)-Grammatik existiert eine äquivalente LR(1)-Grammatik, die dieselbe Sprache erzeugt
 - Es existieren Grammatiken, die **nicht** LR(k) für alle $k \geq 0$ sind

© Prof. J.C. Freytag, Ph.D.

9.5



Besonderheiten von LR(1)-Grammatiken

- LR(1)-Grammatiken werden oft für die Realisierung eines Parsers genutzt; diese heißen dann LR(1)-Parser
- fast alle kontextfreien Konstrukte gängiger Programmiersprachen können in LR(1) ausgedrückt werden

© Prof. J.C. Freytag, Ph.D.

9.6

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



LR-Parsing

Es gibt **drei** (wesentliche) Verfahren zur Generierung der Syntaxanalysetabelle eines LR-Parsers:

1. SLR(1):

- bestimmt von diesen die *kleinste Klasse* von Grammatiken (bzgl. Mächtigkeit)
- kommt mit *kleinsten Tabellen* aus (bzgl. Zustandsanzahl)
- erlaubt eine einfache, schnelle Konstruktion der Tabellen

2. LR(1):

- ist gesamte Klasse der LR(1) Grammatiken (größte Mächtigkeit)
- Erzeugte Tabellen sind sehr groß (bzgl. Zustandsanzahl)
- benötigt *aufwendige* Konstruktion großer Tabellen

3. LALR(1):

- mächtiger als SLR(1), weniger mächtig als LR(1)
- benötigt gleiche Anzahl an Zuständen wie SLR(1)
- erlaubt kanonische Konstruktion, ist aber langsam und erzeugt große Tabellen
- es existieren bessere Konstruktionstechniken als für LR(1)



Vergleich: SLR bzw. LALR gegenüber LR

- Ein LR(1)-Parser für eine gängige Programmiersprache besitzt mehrere **tausend** Zustände
- SLR(1)- bzw. LALR(1)-Parser besitzen für dieselbe Sprache in den meisten Fällen nur mehrere **hundert** Zustände

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



Konstruktion von LR-Syntaxtabellen

Im Weiteren:

- (1) einfaches LR (SLR)
(bestimmt kleinste Menge analysierbarer Grammatiken)
- (2) LR
- (3) LALR



LR(k)-Element (Item)

Die Algorithmen zur Tabellenkonstruktion

- benutzen Mengen von LR(k)-Elementen, auch Konfigurationen genannt, um die möglichen Zustände während des Parsens zu repräsentieren

ein **LR(k)-Element** ist ein Paar $[\alpha, \beta]$, wobei

- α eine Produktion der Grammatik G (z.B. Regelnummer) und " \cdot " eine Markierung in der RS der Regel ist, die anzeigt, wie viel von der RS einer Produktion schon erkannt worden ist;
- β die "look-ahead" Zeichenkette ist, die k Symbole (Terminalsymbole oder "\$") umfasst.

zwei Fälle spielen eine Schlüsselrolle: $k=0$ und $k=1$

- **LR(0)-Element** bei der Konstruktion der Tabelle für **SLR(1)**
- **LR(1)-Element** bei der Konstruktion der **LR(1)**- und **LALR(1)**-Tabellen

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

LR(k)-Element (2)

- Symbol " \bullet " zeigt an, wie viel eines Elements schon in einem Zustand der Erkennung gesehen worden ist:
 - $[A \rightarrow \bullet XYZ]$ zeigt an, dass der Parser eine Zeichenkette sucht, die aus der Zeichenkette XYZ abgeleitet werden kann
 - $[A \rightarrow XY \bullet Z]$ zeigt an, dass der Parser bereits eine Zeichenkette gesehen hat, die aus XY abgeleitet werden konnte und dass er nach einer Zeichenkette sucht, die aus Z abgeleitet werden kann
- LR(0)-Element: kein "Look-ahead"
- Regel $A \rightarrow XYZ$ generiert vier LR(0)-Elemente:
 - $[A \rightarrow \bullet XYZ]$
 - $[A \rightarrow X \bullet YZ]$
 - $[A \rightarrow XY \bullet Z]$
 - $[A \rightarrow XYZ \bullet]$

Regel $A \rightarrow \varepsilon$ generiert nur ein LR(0)-Element:

- $[A \rightarrow \bullet]$

© Prof. J.C. Freytag, Ph.D.

9.11

Konstruktion von SLR-Syntaxtabellen

Idee:

- Konstruktion eines deterministischen endlichen Zustandsautomaten (DFA) für die jeweilige Grammatik, der **gültige Präfixe** erkennt
- Zusammenfassung von Elementen zu Mengen, die Ausgangspunkt der Zustände des SLR-Parsers bilden.
- Elemente könnten auch als Zustände eines Nicht-Deterministischen Automaten angesehen werden, der gültige Vorsilben erkennt (wird hier nicht weiter verfolgt)

© Prof. J.C. Freytag, Ph.D.

9.12

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



Charakteristischer endlicher Automat (CFSM)

- der CFSM für eine Grammatik G ist ein DFA, der alle brauchbaren Präfixe einer Rechtsableitung erkennt:
 - ein brauchbarer (engl. "viable") Präfix ist ein Präfix, der nicht über einen Handle hinweg erweitert werden kann
- der CFSM akzeptiert, wenn eine Handle erkannt worden ist und reduziert werden soll
- um den CFSM zu konstruieren, werden die folgenden Funktionen benötigt:
 - $\text{closure0}(I)$, um die Zustände für ein Element I zu bestimmen
 - $\text{goto0}(I, X)$, um eine Transitionsfunktion für ein Item I und ein Symbol (Terminal oder Nicht-Terminal) X zu bestimmen

© Prof. J.C. Freytag, Ph.D.

9.13



Hüllenoperation: closure0

Sei I eine Menge von Elementen für eine Grammatik G , dann ist die Hülle $\text{closure0}(I)$ die Menge von Elementen, die aus I nach folgenden Regeln konstruiert wird:

- jedes Element von I wird der Hülle $\text{closure0}(I)$ hinzugefügt
- wenn $[A \rightarrow \alpha \cdot B\beta]$ zur Hülle gehört und $B \rightarrow \gamma$ eine Produktion, dann füge das Element $[B \rightarrow \cdot \gamma]$ ebenfalls der Hülle zu

Bemerkung:

D.h., falls der Parser einen brauchbaren Präfix α im Keller gespeichert hat, dann sollte sich die Eingabe zu $B\beta$ reduzieren (oder zu γ für ein anderes Element $[B \rightarrow \cdot \gamma]$ in der Hülle von $[A \rightarrow \alpha \cdot B\beta]$)

© Prof. J.C. Freytag, Ph.D.

9.14

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Berechnung der Hülle

```
function closure0 (I)
  J := I;
  repeat
    for jedes Element  $[A \rightarrow \alpha \cdot B \beta] \in J$  und
      jede Produktion  $B \rightarrow \gamma \in G$  mit  $[B \rightarrow \cdot \gamma] \notin J$ 
    do add  $[B \rightarrow \cdot \gamma]$  to J
  until (no more items can be added to J)
  return J
end_function
```

© Prof. J.C. Freytag, Ph.D.

9.15

closure0 - Beispiel

Grammatik

1	$S \rightarrow E\$$
2	$E \rightarrow E + T$
3	$\quad \mid T$
4	$T \rightarrow \text{id}$
5	$\quad \mid (E)$

Beispiele

- $\text{closure0}([S \rightarrow \cdot E\$]) = \{ [S \rightarrow \cdot E\$], [E \rightarrow \cdot E + T], [E \rightarrow \cdot T], [T \rightarrow \cdot \text{id}], [T \rightarrow \cdot (E)] \}$
- $\text{closure0}([E \rightarrow E + \cdot T]) = \{ [E \rightarrow E + \cdot T], [T \rightarrow \cdot \text{id}], [T \rightarrow \cdot (E)] \}$

Idee der closure0-Funktion:

- "•" gibt an,
 - was als nächstes (rechts von ihm) bei der Erkennung erwartet wird und
 - woher es kommt (durch die Hüllenberechnung)

© Prof. J.C. Freytag, Ph.D.

9.16



Sprungoperation: goto0

Definition:


Sei I eine Menge an LR(0)-Elementen und X ein Grammatiksymbol, dann ist $GOTO(I, X)$ die Hülle der Menge aller Elemente

$[A \rightarrow \alpha X \bullet \beta]$ mit $[A \rightarrow \alpha \bullet X \beta] \in I$

Intuition:

Falls I die Menge aller brauchbaren Elemente für einen brauchbaren Präfix γ ist, dann ist $GOTO(I, X)$ die Menge aller erlaubten (engl. valid) Elemente für den brauchbaren Präfix γX

anschaulich: $GOTO(I, X)$ repräsentiert den (Folge-)Zustand, nachdem X im Zustand I erkannt worden ist



goto0 - Beispiel

Beispiel

sei I die Menge $\{[E' \rightarrow E \bullet], [E \rightarrow E \bullet + T], [E \rightarrow \bullet E + T]\}$,

dann besteht $goto0(I, +)$ aus

- $[E \rightarrow E + \bullet T]$
und davon die Hülle
- $[T \rightarrow \bullet T * F]$
- $[T \rightarrow \bullet F]$
- $[F \rightarrow \bullet (E)]$
- $[F \rightarrow \bullet id]$



Berechnung des Sprungs

```
function goto0 (I,X)
  sei L die Menge aller Items  $[A \rightarrow \alpha X \cdot \beta]$  mit  $[A \rightarrow \alpha \cdot X \beta] \in I$ ;
  return (closure0 (L))
end_function
```



Konstruktion von SLR-Syntaxtabellen

Idee:

- Konstruktion eines Deterministischen Endlichen Zustandsautomaten für die jeweilige Grammatik, der gültige Vorsilben (Präfixe) erkennt
- Zusammenfassung von Elementen zu Mengen, die Ausgangspunkt der Zustände des SLR-Parsers bilden.
- Elemente können als Zustände eines Nicht-Deterministischen Automaten angesehen werden, der gültige Vorsilben erkennt

kanonische LR(0)-Sammlung

- Menge von LR(0)-Elementen, die die Basis bildet zur Konstruktion von SLR-Parsern
- zu deren Konstruktion benötigt man:
 - eine erweiterte Grammatik G' (gegenüber der Ausgangsgrammatik G)
 - zwei Funktionen: Hülle, Sprung

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



Kanonische Kollektion von LR(0)-Elementen

sei G' (eine erweiterte Grammatik von G)

```
function items ( $G'$ )  
   $C \leftarrow \text{closure0} ( \{ [S' \rightarrow \cdot S] \} );$   
  repeat  
    for each set of item  $s \in C$  do  
      for each grammar symbol  $X$  do  
        if  $\text{goto0}(s, X) \neq \emptyset$  and  $\text{goto0}(s, X) \notin C$  then add  
           $\text{goto0}(s, X)$  to  $C$  ;  
  until (no more item sets can be added to  $C$ )  
  return ( $C$ ) ;  
end_function
```

© Prof. J.C. Freytag, Ph.D.

9.21



Beispiel: Kanonische LR(0)-Kollektion

Grammatik

1	$S \rightarrow E \#$
2	$E \rightarrow E + T$
3	$\quad \mid T$
4	$T \rightarrow \text{id}$
5	$\quad \mid (E)$


```
function items ( $G'$ )  
   $C \leftarrow \text{closure0} ( \{ [S' \rightarrow \cdot S] \} );$   
  repeat  
    for each set of item  $s \in C$  do  
      for each grammar symbol  $X$  do  
        if  $\text{goto0}(s, X) \neq \emptyset$  and  $\text{goto0}(s, X) \notin C$  then add  
           $\text{goto0}(s, X)$  to  $C$  ;  
  until (no more item sets can be added to  $C$ )  
  return ( $C$ ) ;  
end_function
```

© Prof. J.C. Freytag, Ph.D.

9.22

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)




Beispiel: Kanonische LR(0)-Kollektion

I_0 :

- $S \rightarrow \cdot E \#$
- $E \rightarrow \cdot E + T$
- $E \rightarrow \cdot T$
- $T \rightarrow \cdot id$
- $T \rightarrow \cdot (E)$

`closure0 ({[S→ ·E]});`

© Prof. J.C. Freytag, Ph.D. 9.23



Beispiel: Kanonische LR(0)-Kollektion

I_0 :

- $S \rightarrow \cdot E \#$
- $E \rightarrow \cdot E + T$
- $E \rightarrow \cdot T$
- $T \rightarrow \cdot id$
- $T \rightarrow \cdot (E)$

E →

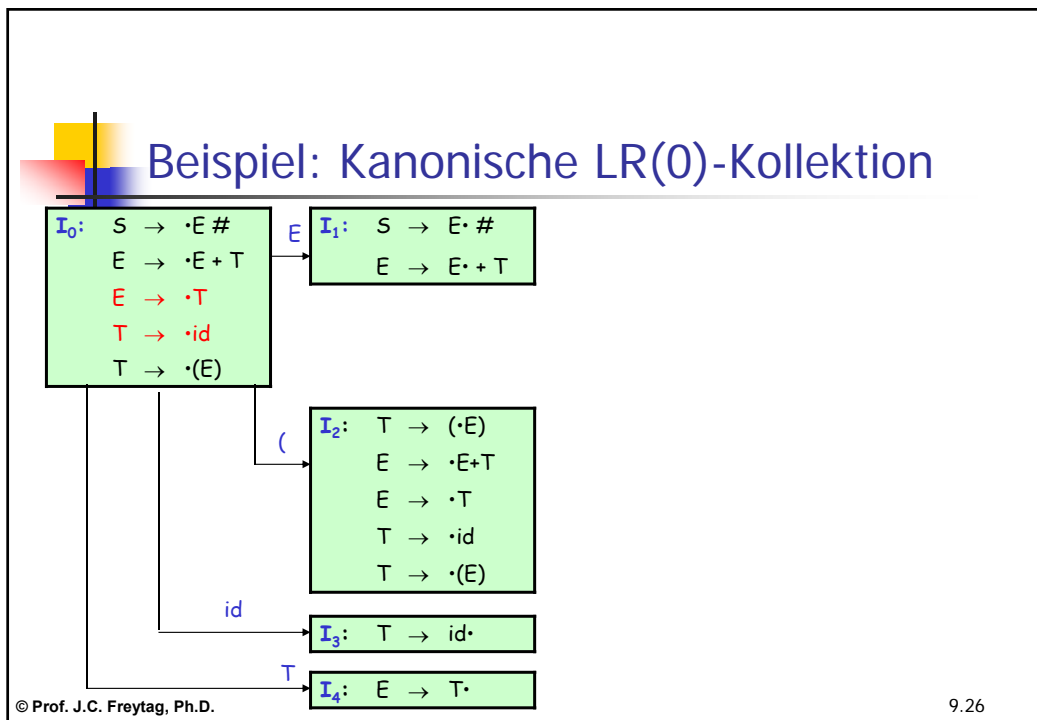
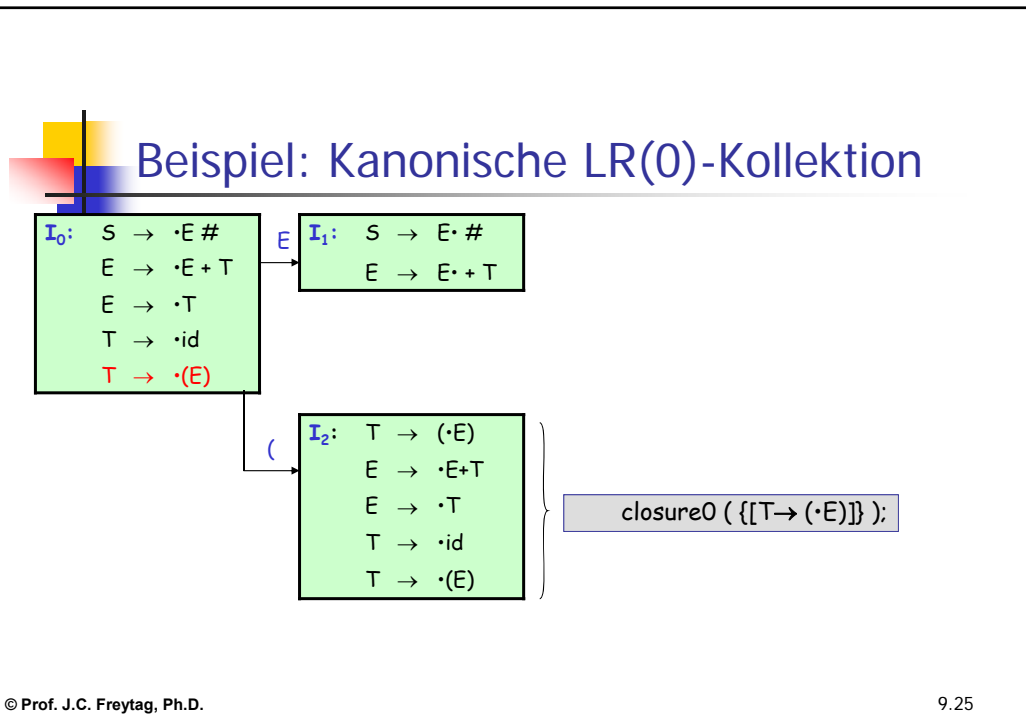
I_1 :

- $S \rightarrow E \cdot \#$
- $E \rightarrow E \cdot + T$

© Prof. J.C. Freytag, Ph.D. 9.24

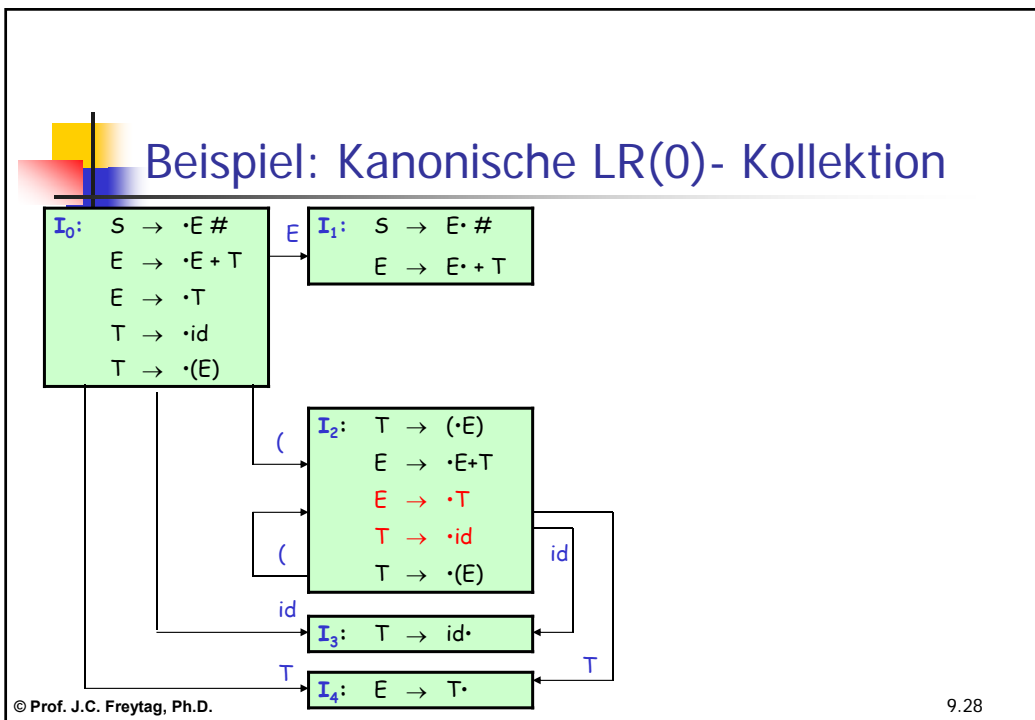
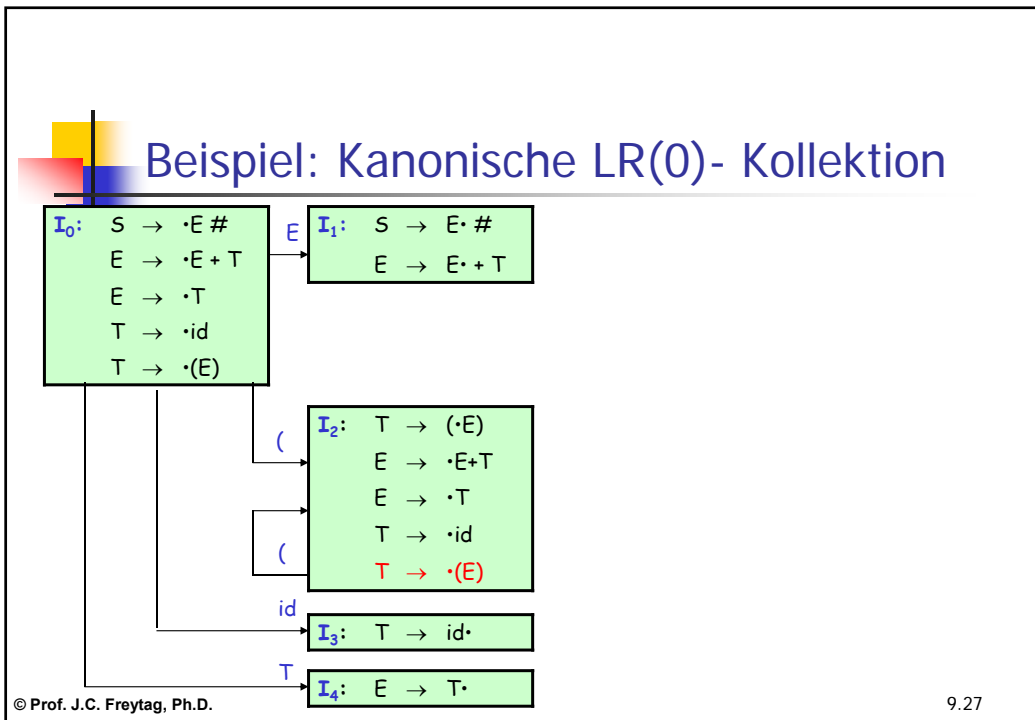
Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



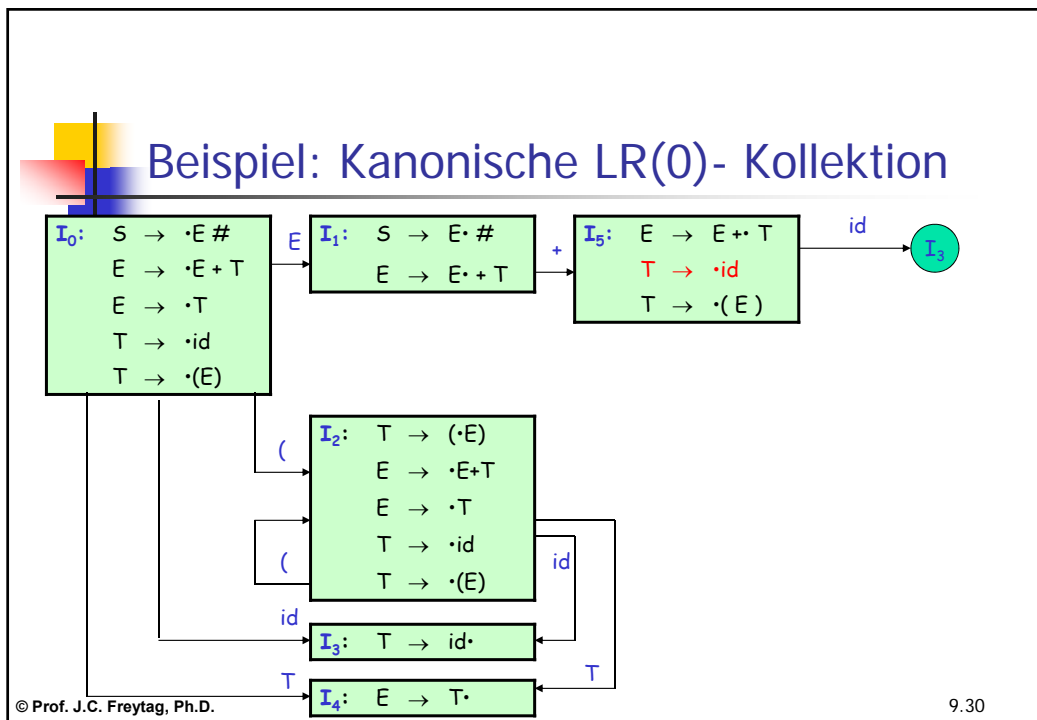
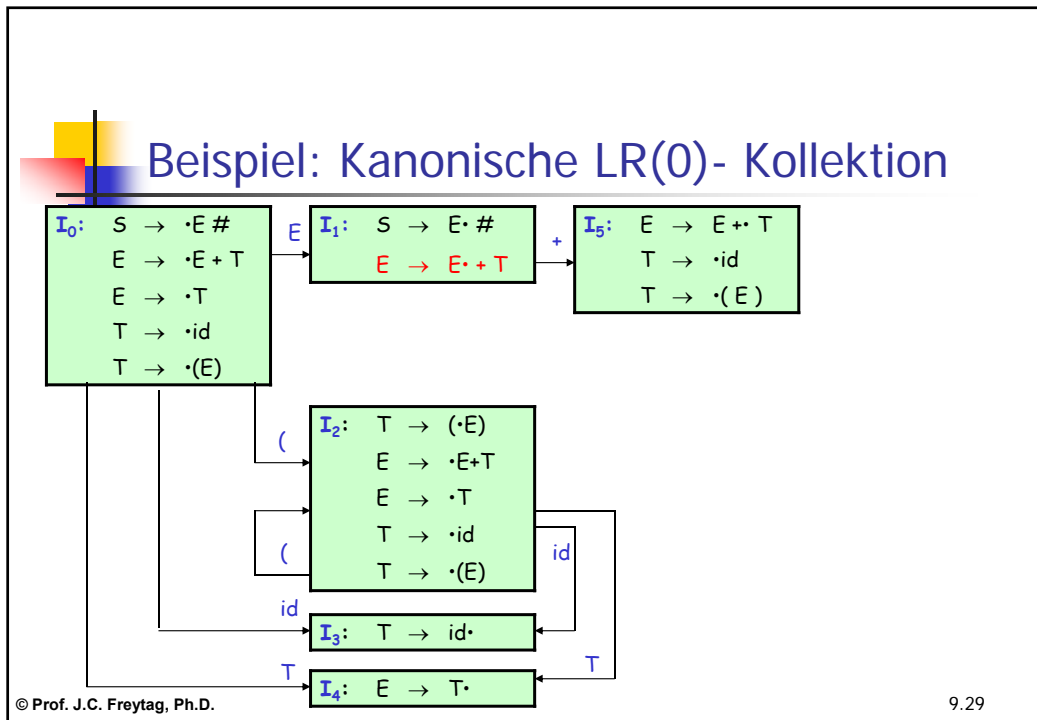
Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



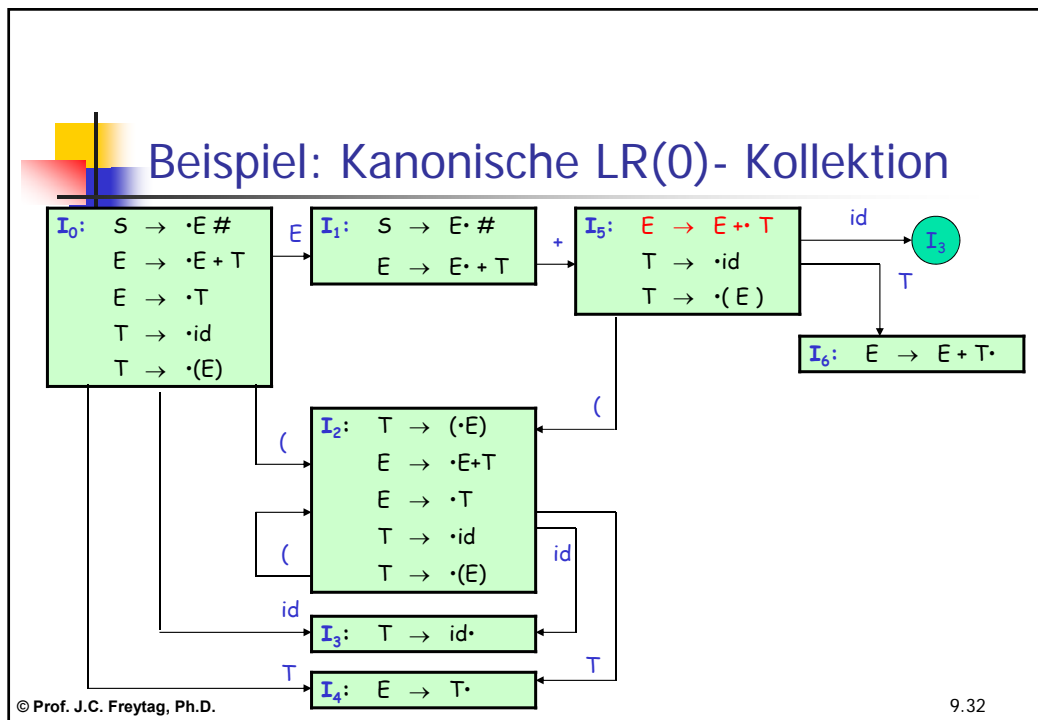
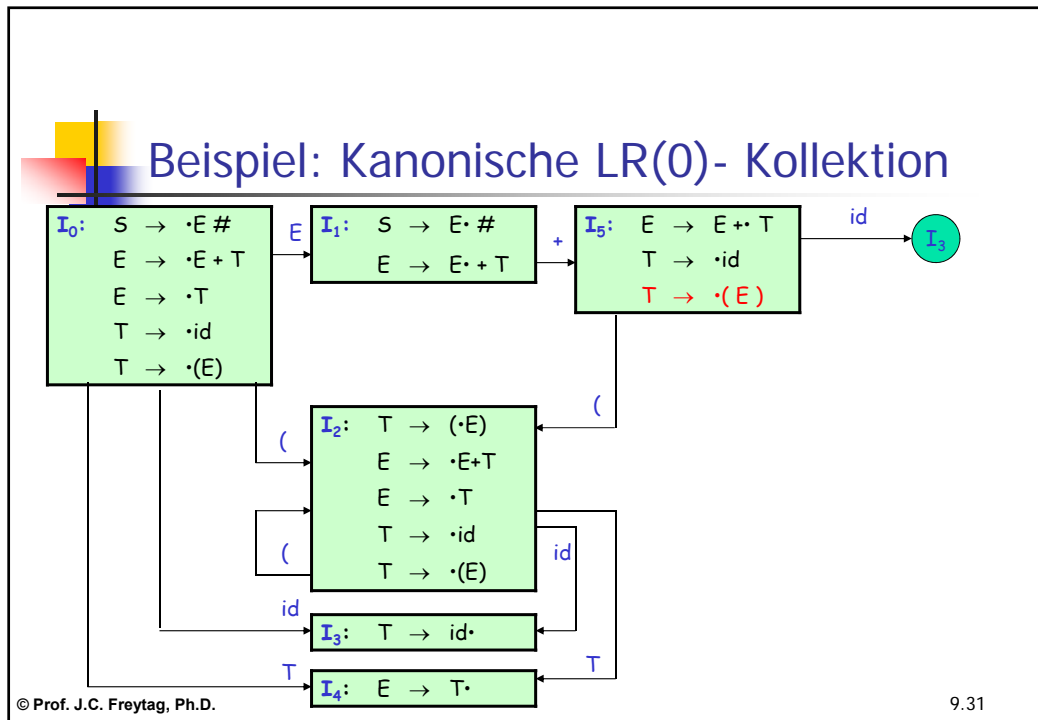
Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



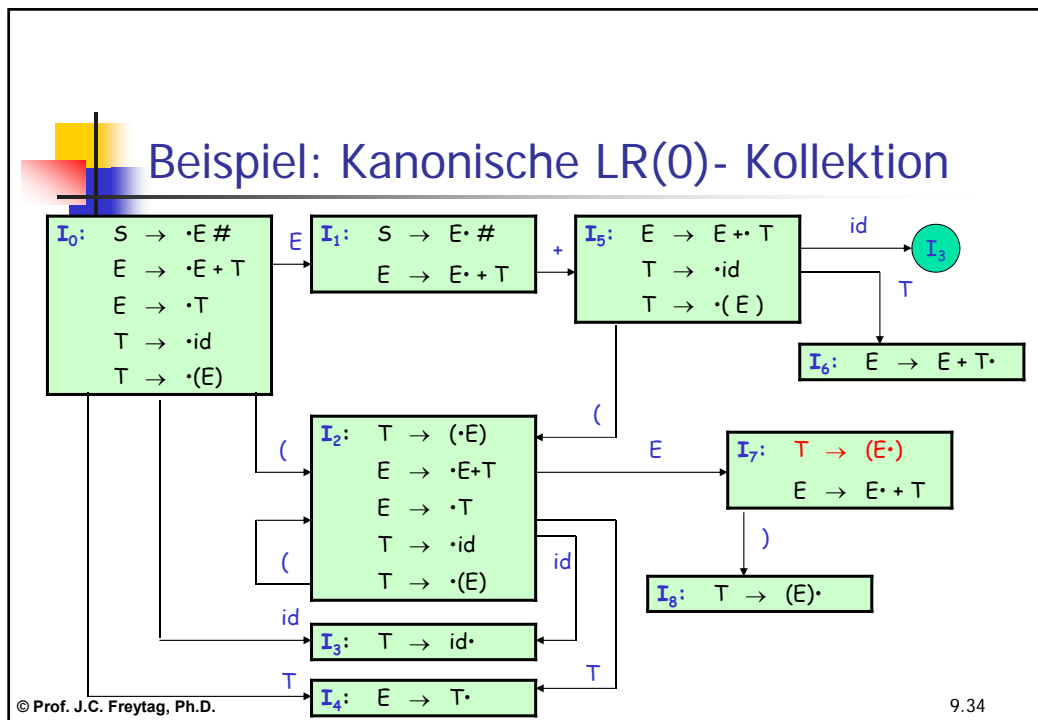
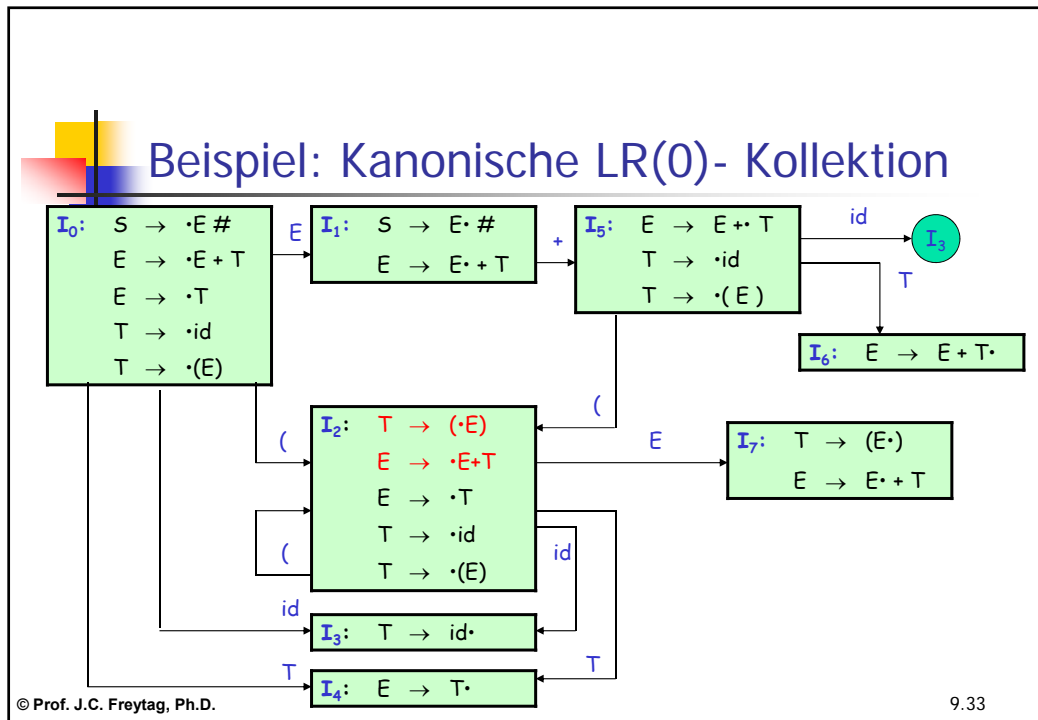
Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



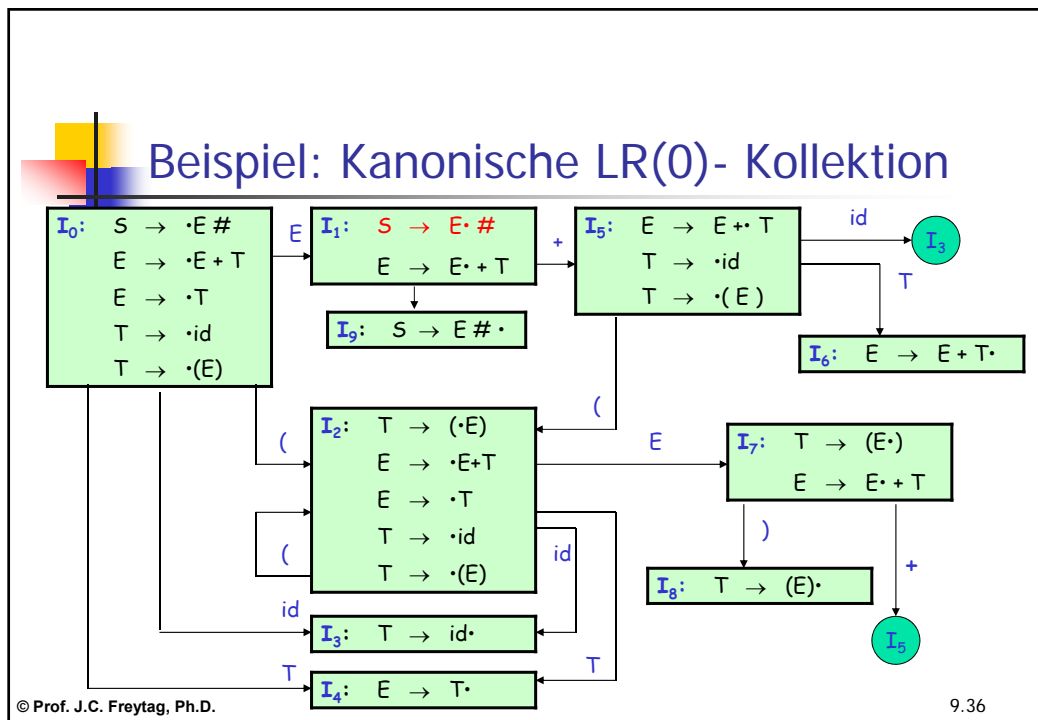
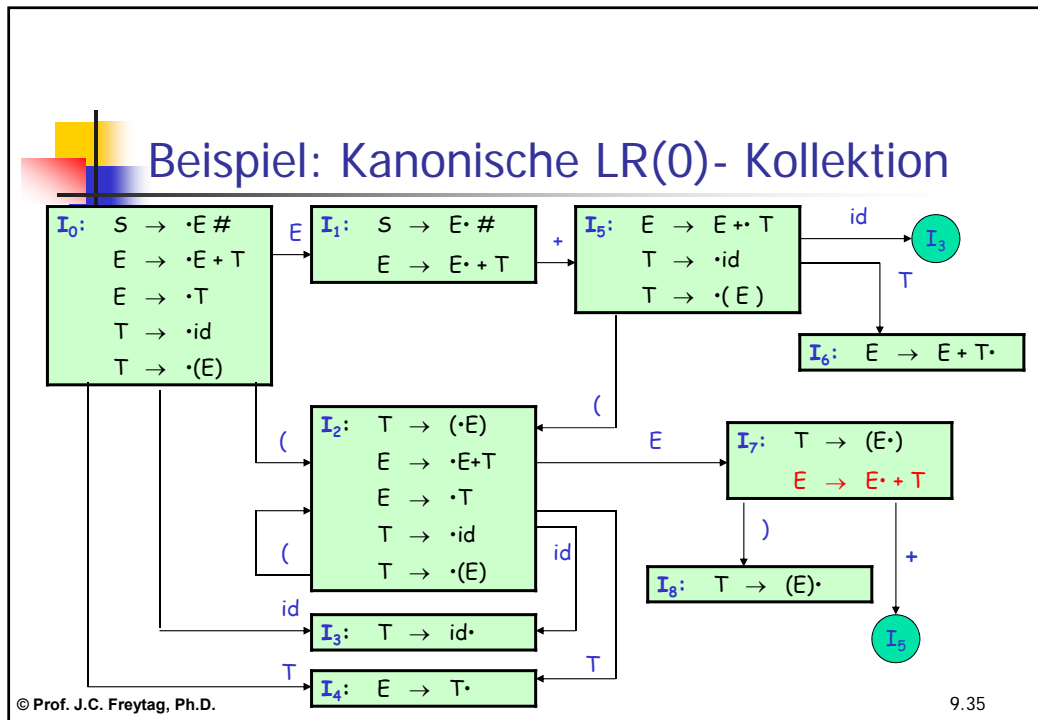
Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



Vorlesung Compilerbau (SoSe 2018)

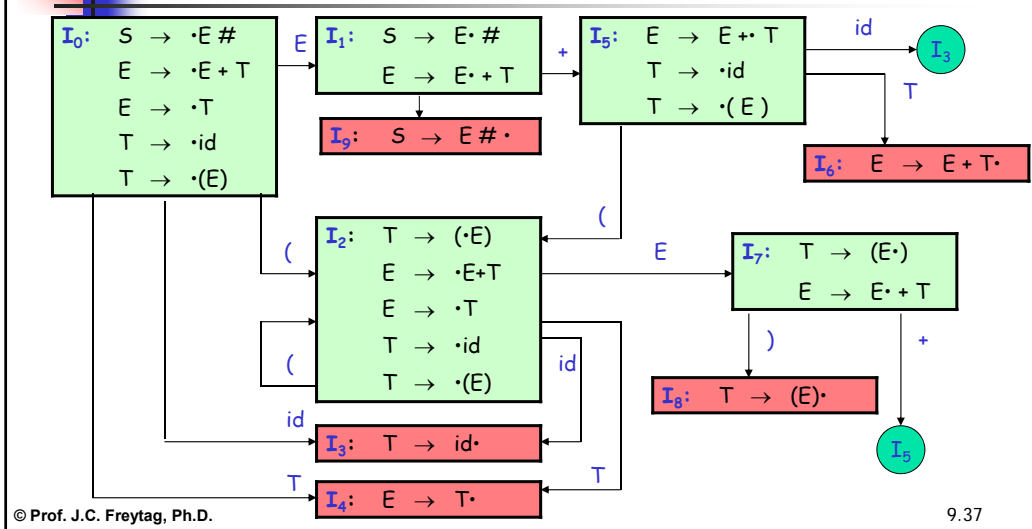
Teil 9: LR/LR(k)-Parsing (2)



Vorlesung Compilerbau (SoSe 2018)

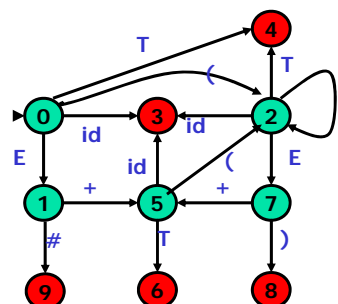
Teil 9: LR/LR(k)-Parsing (2)

Beispiel: Kanonische LR(0)- Kollektion



Beispiel: Kanonische LR(0)- Kollektion

Grammatik	
1	$S \rightarrow E \#$
2	$E \rightarrow E + T$
3	$ T$
4	$T \rightarrow id$
5	$ (E)$



Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Konstruktionsvorschrift für eine SLR(0)-Syntaxtabelle

- Konstruiere die **Mengenkollektion** $C = \{I_0, I_1, \dots, I_n\}$ von LR(0)-Elementen für G
- der **Zustand** i des CFSM wird von I_i wie folgt konstruiert:
 - (a) $[A \rightarrow \alpha \cdot a \beta] \in I_i$ und $\text{goto0}(I_i, a) = I_j$, dann $\text{ACTION}[i, a] \leftarrow \text{"shift } j\text{"}$
 a muss dabei ein Terminal sein (und $\neq \$$)
 - (b) $[A \rightarrow \alpha \cdot] \in I_i$ und $A \neq S'$, dann $\text{ACTION}[i, b] \leftarrow \text{"reduce } A \rightarrow \alpha\text{"}$ für alle Terminale b
 - (c) $[S' \rightarrow S \cdot \$] \in I_i$, dann $\text{ACTION}[i, \$] \leftarrow \text{"accept"}$
- wenn $\text{goto0}(I_i, A) = I_j$ dann $\text{GOTO}[i, A] \leftarrow j$ für alle Nichtterminale A
- setze undefinierte Einträge in **ACTION** und **GOTO** auf "error"
- **Anfangszustand** des Parsers s_0 ist $\text{closure0}([S' \rightarrow \cdot S])$

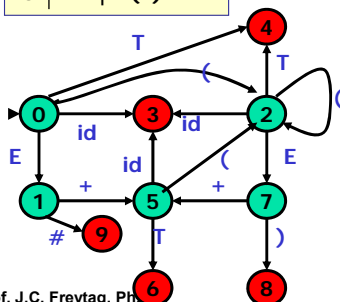
© Prof. J.C. Freytag, Ph.D.

9.39

Beispiel: Konstr. der SLR(0)-Syntaxtabelle

Grammatik

- 1 $S \rightarrow E \#$
- 2 $E \rightarrow E + T$
- 3 $\quad \mid T$
- 4 $T \rightarrow \text{id}$
- 5 $\quad \mid (E)$



Zust	ACTION					GOTO		
	id	(+	#		S	E	T
0	-	-	-	-	-	-	-	-
1	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-

© Prof. J.C. Freytag, Ph.D.

9.40

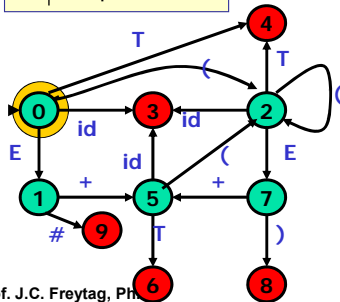
Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Beispiel: Konstr. der SLR(0)-Syntaxtabelle

Grammatik

- 1 $S \rightarrow E \#$
- 2 $E \rightarrow E + T$
- 3 $\quad | T$
- 4 $T \rightarrow id$
- 5 $\quad | (E)$



Zust	ACTION					GOTO		
	id	()	+	#	S	E	T
0	s3	s2	-	-	-	-	1	4
1	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-

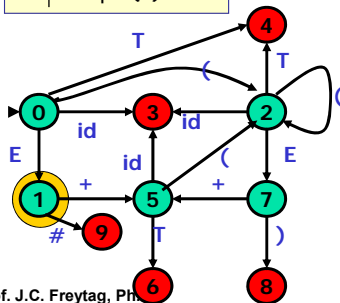
© Prof. J.C. Freytag, PH

9.41

Beispiel: Konstr. der SLR(0)-Syntaxtabelle

Grammatik

- 1 $S \rightarrow E \#$
- 2 $E \rightarrow E + T$
- 3 $\quad | T$
- 4 $T \rightarrow id$
- 5 $\quad | (E)$



Zust	ACTION					GOTO		
	id	()	+	#	S	E	T
0	s3	s2	-	-	-	-	1	4
1	-	-	-	s5	acc	-	-	-
2	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-

© Prof. J.C. Freytag, PH

9.42

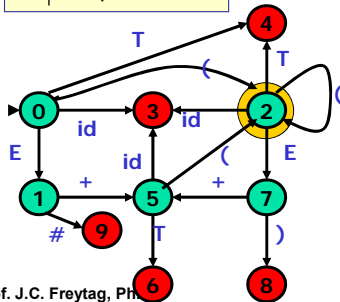
Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Beispiel: Konstr. der SLR(0)-Syntaxtabelle

Grammatik

- 1 $S \rightarrow E \#$
- 2 $E \rightarrow E + T$
- 3 $\quad \mid T$
- 4 $T \rightarrow id$
- 5 $\quad \mid (E)$



Zust	ACTION					GOTO		
	id	()	+	#	S	E	T
0	s3	s2	-	-	-	-	1	4
1	-	-	-	s5	acc	-	-	-
2	s3	s2	-	-	-	-	7	4
3	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-

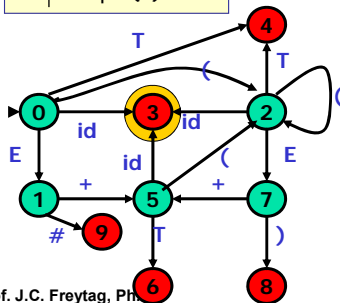
© Prof. J.C. Freytag, PH

9.43

Beispiel: Konstr. der SLR(0)-Syntaxtabelle

Grammatik

- 1 $S \rightarrow E \#$
- 2 $E \rightarrow E + T$
- 3 $\quad \mid T$
- 4 $T \rightarrow id$
- 5 $\quad \mid (E)$



Zust	ACTION					GOTO		
	id	()	+	#	S	E	T
0	s3	s2	-	-	-	-	1	4
1	-	-	-	s5	acc	-	-	-
2	s3	s2	-	-	-	-	7	4
3	r4	r4	r4	r4	r4	-	-	-
4	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-

© Prof. J.C. Freytag, PH

9.44

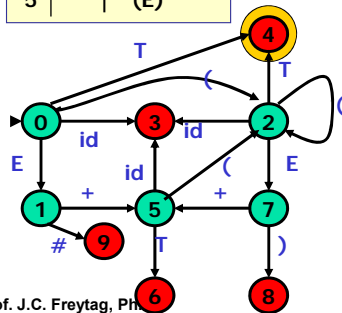
Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Beispiel: Konstr. der SLR(0)-Syntaxtabelle

Grammatik

- 1 $S \rightarrow E \#$
- 2 $E \rightarrow E + T$
- 3 $\quad \mid T$
- 4 $T \rightarrow id$
- 5 $\quad \mid (E)$



Zust	ACTION					GOTO		
	id	()	+	#	S	E	T
0	s3	s2	-	-	-	-	1	4
1	-	-	-	s5	acc	-	-	-
2	s3	s2	-	-	-	-	7	4
3	r4	r4	r4	r4	r4	-	-	-
4	r3	r3	r3	r3	r3	-	-	-
5	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-

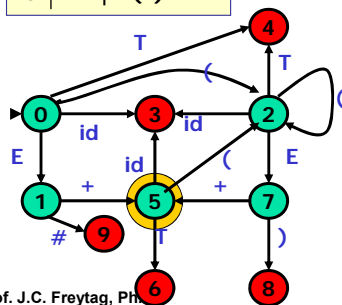
© Prof. J.C. Freytag, PH

9.45

Beispiel: Konstr. der SLR(0)-Syntaxtabelle

Grammatik

- 1 $S \rightarrow E \#$
- 2 $E \rightarrow E + T$
- 3 $\quad \mid T$
- 4 $T \rightarrow id$
- 5 $\quad \mid (E)$



Zust	ACTION					GOTO		
	id	()	+	#	S	E	T
0	s3	s2	-	-	-	-	1	4
1	-	-	-	s5	acc	-	-	-
2	s3	s2	-	-	-	-	7	4
3	r4	r4	r4	r4	r4	-	-	-
4	r3	r3	r3	r3	r3	-	-	-
5	s3	s2	-	-	-	-	-	6
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-

© Prof. J.C. Freytag, PH

9.46

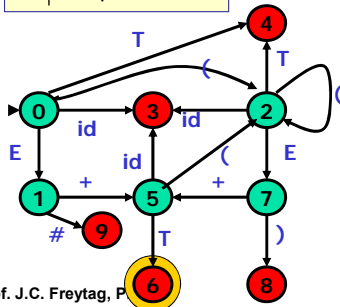
Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Beispiel: Konstr. der SLR(0)-Syntaxtabelle

Grammatik

- 1 $S \rightarrow E \#$
- 2 $E \rightarrow E + T$
- 3 $\quad \mid T$
- 4 $T \rightarrow id$
- 5 $\quad \mid (E)$



Zust	ACTION					GOTO		
	id	()	+	#	S	E	T
0	s3	s2	-	-	-	-	1	4
1	-	-	-	s5	acc	-	-	-
2	s3	s2	-	-	-	-	7	4
3	r4	r4	r4	r4	r4	-	-	-
4	r3	r3	r3	r3	r3	-	-	-
5	s3	s2	-	-	-	-	-	6
6	r2	r2	r2	r2	r2	-	-	-
7	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-

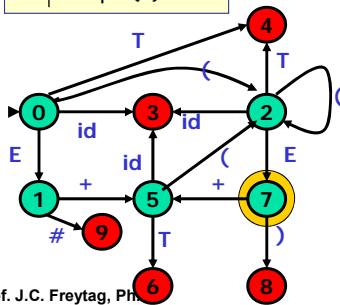
© Prof. J.C. Freytag, P

9.47

Beispiel: Konstr. der SLR(0)-Syntaxtabelle

Grammatik

- 1 $S \rightarrow E \#$
- 2 $E \rightarrow E + T$
- 3 $\quad \mid T$
- 4 $T \rightarrow id$
- 5 $\quad \mid (E)$



Zust	ACTION					GOTO		
	id	()	+	#	S	E	T
0	s3	s2	-	-	-	-	1	4
1	-	-	-	s5	acc	-	-	-
2	s3	s2	-	-	-	-	7	4
3	r4	r4	r4	r4	r4	-	-	-
4	r3	r3	r3	r3	r3	-	-	-
5	s3	s2	-	-	-	-	-	6
6	r2	r2	r2	r2	r2	-	-	-
7	-	-	s8	s5	-	-	-	-
8	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-

© Prof. J.C. Freytag, P

9.48

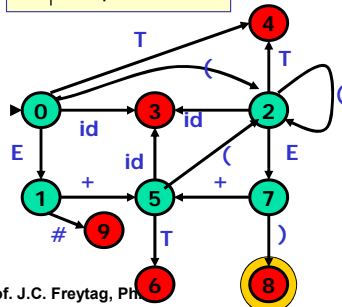
Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Beispiel: Konstr. der SLR(0)-Syntaxtabelle

Grammatik

- 1 $S \rightarrow E \#$
- 2 $E \rightarrow E + T$
- 3 $\quad \mid T$
- 4 $T \rightarrow id$
- 5 $\quad \mid (E)$



Zust	ACTION					GOTO		
	id	()	+	#	S	E	T
0	s3	s2	-	-	-	-	1	4
1	-	-	-	s5	acc	-	-	-
2	s3	s2	-	-	-	-	7	4
3	r4	r4	r4	r4	r4	-	-	-
4	r3	r3	r3	r3	r3	-	-	-
5	s3	s2	-	-	-	-	-	6
6	r2	r2	r2	r2	r2	-	-	-
7	-	-	s8	s5	-	-	-	-
8	r5	r5	r5	r5	r5	-	-	-
9	-	-	-	-	-	-	-	-

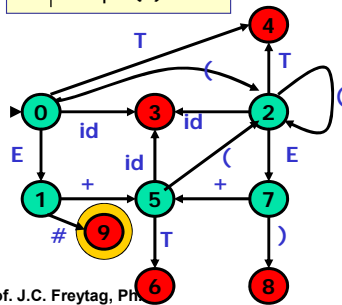
© Prof. J.C. Freytag, PH

9.49

Beispiel: Konstr. der SLR(0)-Syntaxtabelle

Grammatik

- 1 $S \rightarrow E \#$
- 2 $E \rightarrow E + T$
- 3 $\quad \mid T$
- 4 $T \rightarrow id$
- 5 $\quad \mid (E)$



Zust	ACTION					GOTO		
	id	()	+	#	S	E	T
0	s3	s2	-	-	-	-	1	4
1	-	-	-	s5	acc	-	-	-
2	s3	s2	-	-	-	-	7	4
3	r4	r4	r4	r4	r4	-	-	-
4	r3	r3	r3	r3	r3	-	-	-
5	s3	s2	-	-	-	-	-	6
6	r2	r2	r2	r2	r2	-	-	-
7	-	-	s8	s5	-	-	-	-
8	r5	r5	r5	r5	r5	-	-	-
9	np	np	np	np	np	np	np	np

np = "not possible"

© Prof. J.C. Freytag, PH

9.50

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



Mögliche Konflikte in der ACTION-Tabelle

- Falls die LR(0)-Syntaxtabelle Einträge in der ACTION-Tabelle hat, die mehrfach belegt sind, ist die Grammatik **nicht** vom Typ LR(0)
- zwei Konfliktmöglichkeiten
 - **Shift-reduce**: Beides (shift und reduce) ist auf derselben Elementmenge möglich
 - **Reduce-reduce**: Mehr als eine Reduktionsmöglichkeit in der selben Elementmenge
- Konflikte können u.U. durch »Lookahead« in der ACTION-Tabelle gelöst werden
- **Beispiele**:
 - $A \rightarrow \varepsilon \mid a\alpha \Rightarrow$ Shift-Reduce- Konflikt
 - $a := b + c * d \Rightarrow$ erfordert »look-ahead«, um einen Shift-Reduce-Konflikt nach dem Einlesen von c (wegen Präzedenz von * über +) zu verhindern

© Prof. J.C. Freytag, Ph.D.

9.51



Konstruktionsvorschrift für SLR(1)-Syntaxtabelle

- Konstruiere die **Mengenkollektion** $C = \{I_0, I_1, \dots, I_n\}$ von LR(0)-Elementen für G' (Erweiterung von G)
- der **Zustand** i des CFSM wird von I_i wie folgt konstruiert:
 - (a) $[A \rightarrow \alpha \cdot a\beta] \in I_i$ und $\text{goto0}(I_i, a) = I_j$, dann setze **ACTION** $[i, a] \leftarrow$ "shift j"
 a muss dabei ein Terminal sein
 - (b) $[A \rightarrow \alpha \cdot] \in I_i$ und $A \neq S'$, dann setze **ACTION** $[i, a] \leftarrow$ "reduce $A \rightarrow \alpha$ " für alle $a \in \text{FOLLOW}(A)$, A kann nicht S' sein
 - (c) $[S' \rightarrow S\$ \cdot] \in I_i$, dann **ACTION** $[i, \$] \leftarrow$ "accept"
- wenn $\text{goto0}(I_i, A) = I_j$ dann **GOTO** $[i, A] \leftarrow j$ für alle Nichtterminale A
- setze undefinierte Einträge in **ACTION** und **GOTO** auf "error"
- **Anfangszustand** des Parsers s_0 ist $\text{closure0}([S' \rightarrow \cdot S\$])$

© Prof. J.C. Freytag, Ph.D.

9.52

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Beispiel: Kanonische SLR(1)-Kollektion

Grammatik

- 1) $S' \rightarrow E$
- 2) $E \rightarrow E + T$
- 3) $\quad | T$
- 4) $T \rightarrow T * F$
- 5) $\quad | F$
- 6) $F \rightarrow (E)$
- 7) $\quad | id$

	FOLLOW
E	{+, , }
T	{+, *, , }
F	{+, *, , }

I₀
 $S' \rightarrow \cdot E$
 $E \rightarrow \cdot E + T$
 $E \rightarrow \cdot T$
 $T \rightarrow \cdot T * F$
 $T \rightarrow \cdot F$
 $F \rightarrow \cdot (E)$
 $F \rightarrow \cdot id$

I₁
 $S' \rightarrow E \cdot$
 $E \rightarrow E \cdot + T$

I₂
 $E \rightarrow T \cdot$
 $T \rightarrow T \cdot * F$

I₃
 $T \rightarrow F \cdot$

I₄
 $F \rightarrow (\cdot E)$
 $E \rightarrow \cdot E + T$
 $E \rightarrow \cdot T$
 $T \rightarrow \cdot T * F$
 $T \rightarrow \cdot F$
 $F \rightarrow \cdot (E)$
 $F \rightarrow \cdot id$

I₅
 $F \rightarrow id \cdot$

I₆
 $E \rightarrow E + \cdot T$
 $F \rightarrow \cdot T * F$
 $T \rightarrow \cdot F$
 $F \rightarrow \cdot (E)$
 $F \rightarrow \cdot id$

I₇
 $T \rightarrow T * \cdot F$
 $F \rightarrow \cdot (E)$
 $T \rightarrow \cdot id$

I₈
 $F \rightarrow (E \cdot)$
 $E \rightarrow E \cdot + T$

I₉
 $E \rightarrow E + T \cdot$
 $T \rightarrow T \cdot * F$

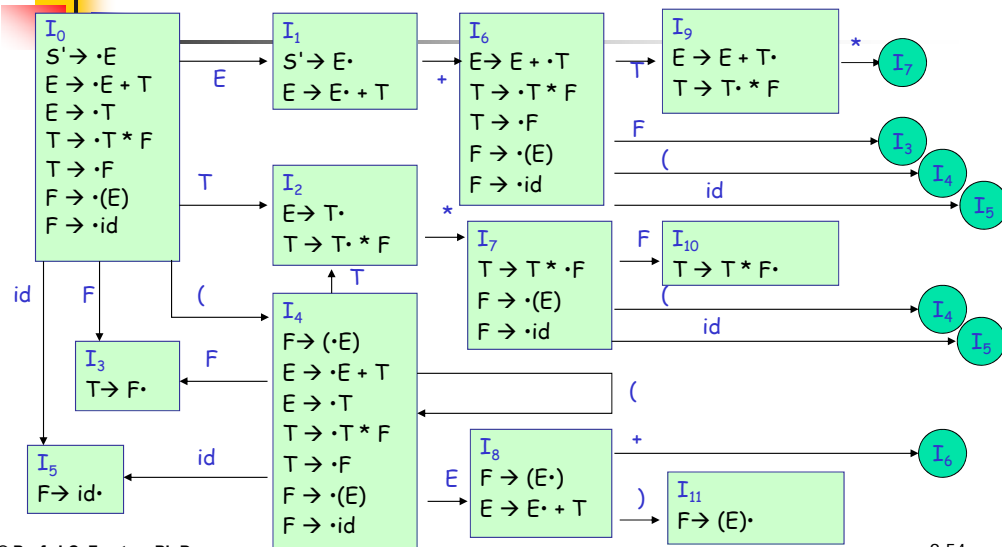
I₁₀
 $T \rightarrow T * F \cdot$

I₁₁
 $F \rightarrow (E) \cdot$

© Prof. J.C. Freytag, Ph.D.

9.53

Beispiel: Automat erkennt gültige Vorsilben



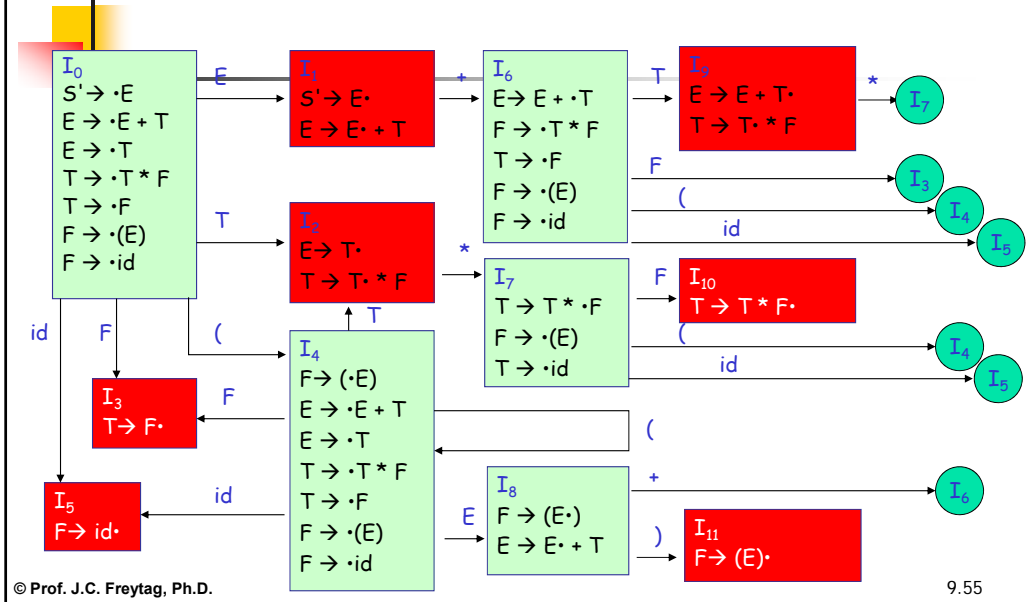
© Prof. J.C. Freytag, Ph.D.

9.54

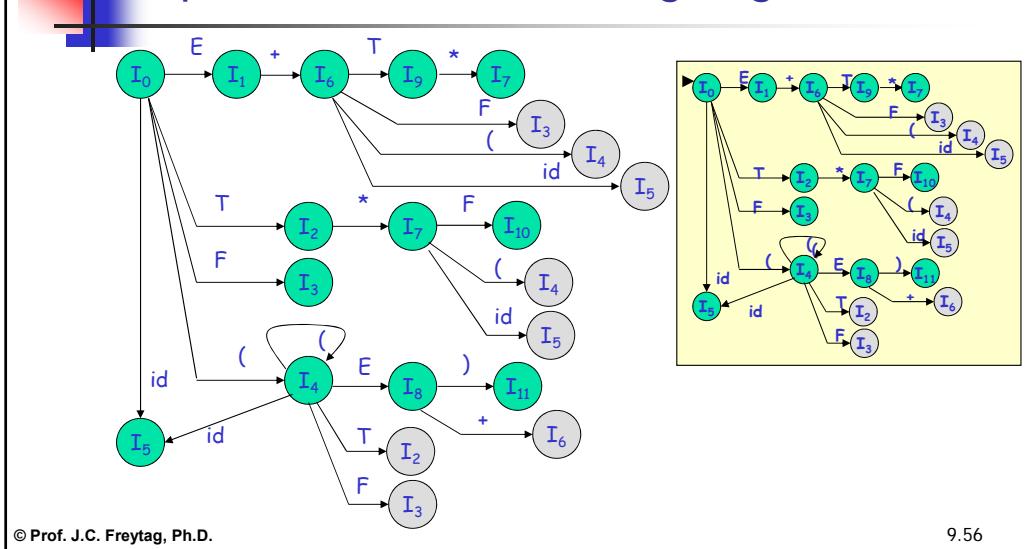
Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Beispiel: Automat erkennt gültige Vorsilben



Beispiel: Automat erkennt gültige Vorsilben



Vorlesung Compilerbau (SoSe 2018)

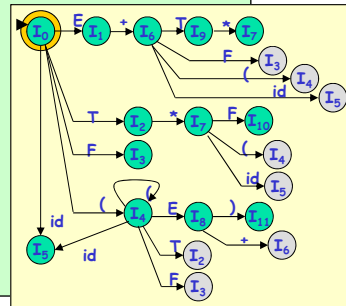
Teil 9: LR/LR(k)-Parsing (2)

Beispiel: Aufbau einer SLR(1)-Syntaxtabelle

Zust	ACTION						GOTO			
	id	+	*	()	\$	S	E	T	F
0	s5	-	-	s4	-	-		1	2	3
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										

The diagram illustrates the LR(0) item sets and transitions for a grammar. The states are represented by circles labeled I0 through I10. I0 is the initial state, marked with a double circle. Transitions are labeled with grammar symbols: E, T, F, id, (, and).

- I0** (Initial State):
 - On **E**, transition to **I1**.
 - On **T**, transition to **I2**.
 - On **F**, transition to **I3**.
 - On **(**, transition to **I4**.
 - On **id**, transition to **I5**.
- I1**:
 - On *****, transition to **I6**.
- I2**:
 - On *****, transition to **I7**.
- I3**:
 - On *****, transition to **I8**.
- I4**:
 - On **E**, transition to **I9**.
 - On **T**, transition to **I10**.
 - On **F**, transition to **I11**.
 - On **(**, transition to **I4** (self-loop).
 - On **id**, transition to **I5**.
- I5**:
 - On **id**, transition to **I5** (self-loop).
- I6**:
 - On **)**, transition to **I0**.
- I7**:
 - On **)**, transition to **I0**.
- I8**:
 - On **)**, transition to **I0**.
- I9**:
 - On **)**, transition to **I0**.
- I10**:
 - On **)**, transition to **I0**.
- I11**:
 - On **)**, transition to **I0**.

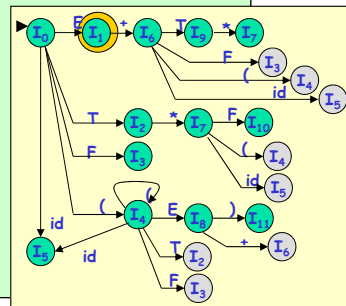


© Prof. J.C. Freytag, Ph.D.

9.57

Beispiel: Aufbau einer SLR(1)-Syntaxtabelle

Zust	ACTION						GOTO		
	id	+	*	()	\$	E	T	F
0	s5	-	-	s4	-	-	1	2	3
1	-	s6	-	-	-	acc	-	-	-
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									



© Prof. J.C. Freytag, Ph.D.

9.58

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Beispiel: Aufbau einer SLR(1)-Syntaxtabelle

ACTION[2,a] ← reduce
für alle $a \in \text{FOLLOW}(E)$

	id	+	*	()	N	GOTO
							E
0	s5	-	-	s4	-	-	1
1	-	s6	-	-	-	-	-
2	-	r3	s7	-	-	r3	-
3	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-

ACTION[2,*] ← shift

Grammatik

- 1) $S' \rightarrow E\$$
- 2) $E \rightarrow E + T$
- 3) $\mid T$
- 4) $T \rightarrow T * F$
- 5) $\mid F$
- 6) $F \rightarrow (E)$
- 7) $\mid \text{id}$

FOLLOW

	FOLLOW
E	+,)
T	+, *,)
F	+, *,)

LR(0) Item Sets and Transitions:

```

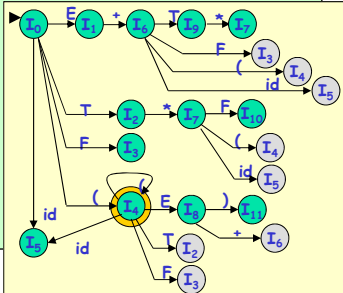
graph LR
    I0["I0: E → T·"] -- "+" --> I1["I1: E → T +·"]
    I0 -- "(" --> I2["I2: E → T (·"]
    I0 -- ")" --> I3["I3: E → T )·"]
    I0 -- "id" --> I4["I4: E → T id·"]
    I1 -- "+" --> I5["I5: E → T + +·"]
    I1 -- "(" --> I6["I6: E → T + (·"]
    I1 -- ")" --> I7["I7: E → T + )·"]
    I1 -- "id" --> I8["I8: E → T + id·"]
    I2 -- "(" --> I9["I9: E → T ( (·"]
    I2 -- ")" --> I10["I10: E → T ( )·"]
    I2 -- "id" --> I11["I11: E → T ( id·"]
    I3 -- "+" --> I12["I12: E → T ) +·"]
    I3 -- "(" --> I13["I13: E → T ) (·"]
    I3 -- ")" --> I14["I14: E → T ) )·"]
    I3 -- "id" --> I15["I15: E → T ) id·"]
    I4 -- "+" --> I16["I16: E → T id +·"]
    I4 -- "(" --> I17["I17: E → T id (·"]
    I4 -- ")" --> I18["I18: E → T id )·"]
    I4 -- "id" --> I19["I19: E → T id id·"]
    I5 -- "+" --> I20["I20: E → T + + +·"]
    I5 -- "(" --> I21["I21: E → T + + (·"]
    I5 -- ")" --> I22["I22: E → T + + )·"]
    I5 -- "id" --> I23["I23: E → T + + id·"]
    I6 -- "+" --> I24["I24: E → T + ( +·"]
    I6 -- "(" --> I25["I25: E → T + ( (·"]
    I6 -- ")" --> I26["I26: E → T + ( )·"]
    I6 -- "id" --> I27["I27: E → T + ( id·"]
    I7 -- "+" --> I28["I28: E → T + ) +·"]
    I7 -- "(" --> I29["I29: E → T + ) (·"]
    I7 -- ")" --> I30["I30: E → T + ) )·"]
    I7 -- "id" --> I31["I31: E → T + ) id·"]
    I8 -- "+" --> I32["I32: E → T + id +·"]
    I8 -- "(" --> I33["I33: E → T + id (·"]
    I8 -- ")" --> I34["I34: E → T + id )·"]
    I8 -- "id" --> I35["I35: E → T + id id·"]
    I9 -- "(" --> I36["I36: E → T ( ( (·"]
    I9 -- ")" --> I37["I37: E → T ( ( )·"]
    I9 -- "id" --> I38["I38: E → T ( ( id·"]
    I10 -- "+" --> I39["I39: E → T ( ) +·"]
    I10 -- "(" --> I40["I40: E → T ( ) (·"]
    I10 -- ")" --> I41["I41: E → T ( ) )·"]
    I10 -- "id" --> I42["I42: E → T ( ) id·"]
    I11 -- "+" --> I43["I43: E → T ( id +·"]
    I11 -- "(" --> I44["I44: E → T ( id (·"]
    I11 -- ")" --> I45["I45: E → T ( id )·"]
    I11 -- "id" --> I46["I46: E → T ( id id·"]
    I12 -- "+" --> I47["I47: E → T ) + +·"]
    I12 -- "(" --> I48["I48: E → T ) + (·"]
    I12 -- ")" --> I49["I49: E → T ) + )·"]
    I12 -- "id" --> I50["I50: E → T ) + id·"]
    I13 -- "(" --> I51["I51: E → T ) ( (·"]
    I13 -- ")" --> I52["I52: E → T ) ( )·"]
    I13 -- "id" --> I53["I53: E → T ) ( id·"]
    I14 -- "+" --> I54["I54: E → T ) ) +·"]
    I14 -- "(" --> I55["I55: E → T ) ) (·"]
    I14 -- ")" --> I56["I56: E → T ) ) )·"]
    I14 -- "id" --> I57["I57: E → T ) ) id·"]
    I15 -- "+" --> I58["I58: E → T ) id +·"]
    I15 -- "(" --> I59["I59: E → T ) id (·"]
    I15 -- ")" --> I60["I60: E → T ) id )·"]
    I15 -- "id" --> I61["I61: E → T ) id id·"]
    I16 -- "+" --> I62["I62: E → T + id + +·"]
    I16 -- "(" --> I63["I63: E → T + id + (·"]
    I16 -- ")" --> I64["I64: E → T + id + )·"]
    I16 -- "id" --> I65["I65: E → T + id + id·"]
    I17 -- "+" --> I66["I66: E → T + id ( +·"]
    I17 -- "(" --> I67["I67: E → T + id ( (·"]
    I17 -- ")" --> I68["I68: E → T + id ( )·"]
    I17 -- "id" --> I69["I69: E → T + id ( id·"]
    I18 -- "+" --> I70["I70: E → T + id ) +·"]
    I18 -- "(" --> I71["I71: E → T + id ) (·"]
    I18 -- ")" --> I72["I72: E → T + id ) )·"]
    I18 -- "id" --> I73["I73: E → T + id ) id·"]
    I19 -- "+" --> I74["I74: E → T + id id +·"]
    I19 -- "(" --> I75["I75: E → T + id id (·"]
    I19 -- ")" --> I76["I76: E → T + id id )·"]
    I19 -- "id" --> I77["I77: E → T + id id id·"]
    I20 -- "+" --> I78["I78: E → T + + + +·"]
    I20 -- "(" --> I79["I79: E → T + + + (·"]
    I20 -- ")" --> I80["I80: E → T + + + )·"]
    I20 -- "id" --> I81["I81: E → T + + + id·"]
    I21 -- "+" --> I82["I82: E → T + + ( +·"]
    I21 -- "(" --> I83["I83: E → T + + ( (·"]
    I21 -- ")" --> I84["I84: E → T + + ( )·"]
    I21 -- "id" --> I85["I85: E → T + + ( id·"]
    I22 -- "+" --> I86["I86: E → T + + ) +·"]
    I22 -- "(" --> I87["I87: E → T + + ) (·"]
    I22 -- ")" --> I88["I88: E → T + + ) )·"]
    I22 -- "id" --> I89["I89: E → T + + ) id·"]
    I23 -- "+" --> I90["I90: E → T + + id +·"]
    I23 -- "(" --> I91["I91: E → T + + id (·"]
    I23 -- ")" --> I92["I92: E → T + + id )·"]
    I23 -- "id" --> I93["I93: E → T + + id id·"]
    I24 -- "+" --> I94["I94: E → T + ( + +·"]
    I24 -- "(" --> I95["I95: E → T + ( + (·"]
    I24 -- ")" --> I96["I96: E → T + ( + )·"]
    I24 -- "id" --> I97["I97: E → T + ( + id·"]
    I25 -- "+" --> I98["I98: E → T + ( ( +·"]
    I25 -- "(" --> I99["I99: E → T + ( ( (·"]
    I25 -- ")" --> I100["I100: E → T + ( ( )·"]
    I25 -- "id" --> I101["I101: E → T + ( ( id·"]
    I26 -- "+" --> I102["I102: E → T + ( ) +·"]
    I26 -- "(" --> I103["I103: E → T + ( ) (·"]
    I26 -- ")" --> I104["I104: E → T + ( ) )·"]
    I26 -- "id" --> I105["I105: E → T + ( ) id·"]
    I27 -- "+" --> I106["I106: E → T + ( id +·"]
    I27 -- "(" --> I107["I107: E → T + ( id (·"]
    I27 -- ")" --> I108["I108: E → T + ( id )·"]
    I27 -- "id" --> I109["I109: E → T + ( id id·"]
    I28 -- "+" --> I110["I110: E → T + ) id +·"]
    I28 -- "(" --> I111["I111: E → T + ) id (·"]
    I28 -- ")" --> I112["I112: E → T + ) id )·"]
    I28 -- "id" --> I113["I113: E → T + ) id id·"]
    I29 -- "+" --> I114["I114: E → T + ) ( +·"]
    I29 -- "(" --> I115["I115: E → T + ) ( (·"]
    I29 -- ")" --> I116["I116: E → T + ) ( )·"]
    I29 -- "id" --> I117["I117: E → T + ) ( id·"]
    I30 -- "+" --> I118["I118: E → T + ) ) +·"]
    I30 -- "(" --> I119["I119: E → T + ) ) (·"]
    I30 -- ")" --> I120["I120: E → T + ) ) )·"]
    I30 -- "id" --> I121["I121: E → T + ) ) id·"]
    I31 -- "+" --> I122["I122: E → T + ) id id +·"]
    I31 -- "(" --> I123["I123: E → T + ) id id (·"]
    I31 -- ")" --> I124["I124: E → T + ) id id )·"]
    I31 -- "id" --> I125["I125: E → T + ) id id id·"]
    I32 -- "+" --> I126["I126: E → T + id + + +·"]
    I32 -- "(" --> I127["I127: E → T + id + + (·"]
    I32 -- ")" --> I128["I128: E → T + id + + )·"]
    I32 -- "id" --> I129["I129: E → T + id + + id·"]
    I33 -- "+" --> I130["I130: E → T + id + ( +·"]
    I33 -- "(" --> I131["I131: E → T + id + ( (·"]
    I33 -- ")" --> I132["I132: E → T + id + ( )·"]
    I33 -- "id" --> I133["I133: E → T + id + ( id·"]
    I34 -- "+" --> I134["I134: E → T + id + ) +·"]
    I34 -- "(" --> I135["I135: E → T + id + ) (·"]
    I34 -- ")" --> I136["I136: E → T + id + ) )·"]
    I34 -- "id" --> I137["I137: E → T + id + ) id·"]
    I35 -- "+" --> I138["I138: E → T + id + id +·"]
    I35 -- "(" --> I139["I139: E → T + id + id (·"]
    I35 -- ")" --> I140["I140: E → T + id + id )·"]
    I35 -- "id" --> I141["I141: E → T + id + id id·"]
    I36 -- "+" --> I142["I142: E → T + id ( + +·"]
    I36 -- "(" --> I143["I143: E → T + id ( + (·"]
    I36 -- ")" --> I144["I144: E → T + id ( + )·"]
    I36 -- "id" --> I145["I145: E → T + id ( + id·"]
    I37 -- "+" --> I146["I146: E → T + id ( ) +·"]
    I37 -- "(" --> I147["I147: E → T + id ( ) (·"]
    I37 -- ")" --> I148["I148: E → T + id ( ) )·"]
    I37 -- "id" --> I149["I149: E → T + id ( ) id·"]
    I38 -- "+" --> I150["I150: E → T + id ( id +·"]
    I38 -- "(" --> I151["I151: E → T + id ( id (·"]
    I38 -- ")" --> I152["I152: E → T + id ( id )·"]
    I38 -- "id" --> I153["I153: E → T + id ( id id·"]
    I39 -- "+" --> I154["I154: E → T + ( ) id +·"]
    I39 -- "(" --> I155["I155: E → T + ( ) id (·"]
    I39 -- ")" --> I156["I156: E → T + ( ) id )·"]
    I39 -- "id" --> I157["I157: E → T + ( ) id id·"]
    I40 -- "+" --> I158["I158: E → T + ( id + +·"]
    I40 -- "(" --> I159["I159: E → T + ( id + (·"]
    I40 -- ")" --> I160["I160: E → T + ( id + )·"]
    I40 -- "id" --> I161["I161: E → T + ( id + id·"]
    I41 -- "+" --> I162["I162: E → T + ( ) id id +·"]
    I41 -- "(" --> I163["I163: E → T + ( ) id id (·"]
    I41 -- ")" --> I164["I164: E → T + ( ) id id )·"]
    I41 -- "id" --> I165["I165: E → T + ( ) id id id·"]
    I42 -- "+" --> I166["I166: E → T + ( id id +·"]
    I42 -- "(" --> I167["I167: E → T + ( id id (·"]
    I42 -- ")" --> I168["I168: E → T + ( id id )·"]
    I42 -- "id" --> I169["I169: E → T + ( id id id·"]
    I43 -- "+" --> I170["I170: E → T + ( id + id +·"]
    I43 -- "(" --> I171["I171: E → T + ( id + id (·"]
    I43 -- ")" --> I172["I172: E → T + ( id + id )·"]
    I43 -- "id" --> I173["I173: E → T + ( id + id id·"]
    I44 -- "+" --> I174["I174: E → T + ( id ( + +·"]
    I44 -- "(" --> I175["I175: E → T + ( id ( + (·"]
    I44 -- ")" --> I176["I176: E → T + ( id ( + )·"]
    I44 -- "id" --> I177["I177: E → T + ( id ( + id·"]
    I45 -- "+" --> I178["I178: E → T + ( id ) id +·"]
    I45 -- "(" --> I179["I179: E → T + ( id ) id (·"]
    I45 -- ")" --> I180["I180: E → T + ( id ) id )·"]
    I45 -- "id" --> I181["I181: E → T + ( id ) id id·"]
    I46 -- "+" --> I182["I182: E → T + ( id id id +·"]
    I46 -- "(" --> I183["I183: E → T + ( id id id (·"]
    I46 -- ")" --> I184["I184: E → T + ( id id id )·"]
    I46 -- "id" --> I185["I185: E → T + ( id id id id·"]
    I47 -- "+" --> I186["I186: E → T + ) id id +·"]
    I47 -- "(" --> I187["I187: E → T + ) id id (·"]
    I47 -- ")" --> I188["I188: E → T + ) id id )·"]
    I47 -- "id" --> I189["I189: E → T + ) id id id·"]
    I48 -- "+" --> I190["I190: E → T + ) id ( + +·"]
    I48 -- "(" --> I191["I191: E → T + ) id ( + (·"]
    I48 -- ")" --> I192["I192: E → T + ) id ( + )·"]
    I48 -- "id" --> I193["I193: E → T + ) id ( + id·"]
    I49 -- "+" --> I194["I194: E → T + ) id ) id +·"]
    I49 -- "(" --> I195["I195: E → T + ) id ) id (·"]
    I49 -- ")" --> I196["I196: E → T + ) id ) id )·"]
    I49 -- "id" --> I197["I197: E → T + ) id ) id id·"]
    I50 -- "+" --> I198["I198: E → T + ) id id id +·"]
    I50 -- "(" --> I199["I199: E → T + ) id id id (·"]
    I50 -- ")" --> I200["I200: E → T + ) id id id )·"]
    I50 -- "id" --> I201["I201: E → T + ) id id id id·"]
    I51 -- "+" --> I202["I202: E → T + id ( + + +·"]
    I51 -- "(" --> I203["I203: E → T + id ( + + (·"]
    I51 -- ")" --> I204["I204: E → T + id ( + + )·"]
    I51 -- "id" --> I205["I205: E → T + id ( + + id·"]
    I52 -- "+" --> I206["I206: E → T + id ( ) id +·"]
    I52 -- "(" --> I207["I207: E → T + id ( ) id (·"]
    I52 -- ")" --> I208["I208: E → T + id ( ) id )·"]
    I52 -- "id" --> I209["I209: E → T + id ( ) id id·"]
    I53 -- "+" --> I210["I210: E → T + id ( id id +·"]
    I53 -- "(" --> I211["I211: E → T + id ( id id (·"]
    I53 -- ")" --> I212["I212: E → T + id ( id id )·"]
    I53 -- "id" --> I213["I213: E → T + id ( id id id·"]
    I54 -- "+" --> I214["I214: E → T + ( id + id id +·"]
    I54 -- "(" --> I215["I215: E → T + ( id + id id (·"]
    I54 -- ")" --> I216["I216: E → T + ( id + id id )·"]
    I54 -- "id" --> I217["I217: E → T + ( id + id id id·"]
    I55 -- "+" --> I218["I218: E → T + ( id ) id id id +·"]
    I55 -- "(" --> I219["I219: E → T + ( id ) id id id (·"]
    I55 -- ")" --> I220["I220: E → T + ( id ) id id id )·"]
    I55 -- "id" --> I221["I221: E → T + ( id ) id id id id·"]
    I56 -- "+" --> I222["I222: E → T + ( id id id id +·"]
    I56 -- "(" --> I223["I223: E → T + ( id id id id (·"]
    I56 -- ")" --> I224["I224: E → T + ( id id id id )·"]
    I56 -- "id" --> I225["I225: E → T + ( id id id id id·"]
    I57 -- "+" --> I226["I226: E → T + ( id id id id id +·"]
    I57 -- "(" --> I227["I227: E → T + ( id id id id id (·"]
    I57 -- ")" --> I228["I228: E → T + ( id id id id id )·"]
    I57 -- "id" --> I229["I229: E → T + ( id id id id id id·"]
    I58 -- "+" --> I230["I230: E → T + ) id id id id +·"]
    I58 -- "(" --> I231["I231: E → T + ) id id id id (·"]
    I58 -- ")" --> I232["I232: E → T + ) id id id id )·"]
    I58 -- "id" --> I233["I233: E → T + ) id id id id id·"]
    I59 -- "+" --> I234["I234: E → T + ) id id ( id + +·"]
    I59 -- "(" --> I235["I235: E → T + ) id id ( id + (·"]
    I59 -- ")" --> I236["I236: E → T + ) id id ( id + )·"]
    I59 -- "id" --> I237["I237: E → T + ) id id ( id + id·"]
    I60 -- "+" --> I238["I238: E → T + ) id id ) id id +·"]
    I60 -- "(" --> I239["I239: E → T + ) id id ) id id (·"]
    I60 -- ")" --> I240["I240: E → T + ) id id ) id id )·"]
    I60 -- "id" --> I241["I241: E → T + ) id id ) id id id·"]
    I61 -- "+" --> I242["I242: E → T + ) id id id id id +·"]
    I61 -- "(" --> I243["I243: E → T + ) id id id id id (·"]
    I61 -- ")" --> I244["I244: E → T + ) id id id id id )·"]
    I61 -- "id" --> I245["I245: E → T + ) id id id id id id·"]
    I62 -- "+" --> I246["I246: E → T + id + id + +·"]
    I62 -- "(" --> I247["I247: E → T + id + id + (·"]
    I62 -- ")" --> I248["I248: E → T + id + id + )·"]
    I62 -- "id" --> I249["I249: E → T + id + id + id·"]
    I63 -- "+" --> I250["I250: E → T + id + id ( + +·"]
    I63 -- "(" --> I251["I251: E → T + id + id ( + (·"]
    I63 -- ")" --> I252["I252: E → T + id + id ( + )·"]
    I63 -- "id" --> I253["I253: E → T + id + id ( + id·"]
    I64 -- "+" --> I254["I254: E → T + id + id ) id + +·"]
    I64 -- "(" --> I255["I255: E → T + id + id ) id + (·"]
    I64 -- ")" --> I256["I256: E → T + id + id ) id + )·"]
    I64 -- "id" --> I257["I257: E → T + id + id ) id + id·"]
    I65 -- "+" --> I258["I258: E → T + id + id id + +·"]
    I65 -- "(" --> I259["I259: E → T + id + id id + (·"]
    I65 -- ")" --> I260["I260: E → T + id + id id + )·"]
    I65 -- "id" --> I261["I261: E → T + id + id id + id·"]
    I66 -- "+" --> I262["I262: E → T + id + id ( id + +·"]
    I66 -- "(" --> I263["I263: E → T + id + id ( id + (·"]
    I66 -- ")" --> I264["I264: E → T + id + id ( id + )·"]
    I66 -- "id" --> I265["I265: E → T + id + id ( id + id·"]
    I67 -- "+" --> I266["I266: E → T + id + id ( ) id + +·"]
    I67 -- "(" --> I267["I267: E → T + id + id ( ) id + (·"]
    I67 -- ")" --> I268["I268: E → T + id + id ( ) id + )·"]
    I67 -- "id" --> I269["I269: E → T + id + id ( ) id + id·"]
    I68 -- "+" --> I270["I270: E → T + id + id ( id id + +·"]
    I68 -- "(" --> I271["I271: E → T + id + id ( id id + (·"]
    I68 -- ")" --> I272["I272: E → T + id + id ( id id + )·"]
    I68 -- "id" --> I273["I273: E → T + id + id ( id id + id·"]
    I69 -- "+" --> I274["I274: E → T + id + id ( id id id + +·"]
    I69 -- "(" --> I275["I275: E → T + id + id ( id id id + (·"]
    I69 -- ")" --> I276["I276: E → T + id + id ( id id id + )·"]
    I69 -- "id" --> I277["I277: E → T + id + id ( id id id + id·"]
    I70 -- "+" --> I278["I278: E → T + id + id ( id id ) id + +·"]
    I70 -- "(" --> I279["I279: E → T + id + id ( id id ) id + (·"]
    I70 -- ")" --> I280["I280: E → T + id + id ( id id ) id + )·"]
    I70 -- "id" --> I281["I281: E → T + id + id ( id id ) id + id·"]
    I71 -- "+" --> I282["I282: E → T + id + id ( id id id id + +·"]
    I71 -- "(" --> I283["I283: E → T + id + id ( id id id id + (·"]
    I71 -- ")" --> I284["I284: E → T + id + id ( id id id id + )·"]
    I71 -- "id" --> I285["I285: E → T + id + id ( id id id id + id·"]
    I72 -- "+" --> I286["I286: E → T + id + id ( id id id id id + +·"]
    I72 -- "(" --> I287["I287: E → T + id + id ( id id id id id + (·"]
    I72 -- ")" --> I288["I288: E → T + id + id ( id id id id id + )·"]
    I72 -- "id" --> I289["I289: E → T + id + id ( id id id id id + id·"]
    I73 -- "+" --> I290["I290: E → T + id + id ( id id id id id id + +·"]
    I73 -- "(" --> I291["I291: E → T + id + id ( id id id id id id + (·"]
    I73 -- ")" --> I292["I292: E → T + id + id ( id id id id id id + )·"]
    I73 -- "id" --> I293["I293: E → T + id + id ( id id id id id id + id·"]
    I74 -- "+" --> I294["I294: E → T + id + id ( id id id id id id id + +·"]
    I74 -- "(" --> I295["I295: E → T + id + id ( id id id id id id id + (·"]
    I74 -- ")" --> I296["I296: E → T + id + id ( id id id id id id id + )·"]
    I74 -- "id" --> I297["I297: E → T + id + id ( id id id id id id id + id·"]
    I75 -- "+" --> I298["I298: E → T + id + id ( id id id id id id id id + +·"]
    I75 -- "(" --> I299["I299: E → T + id + id ( id id id id id id id id + (·"]
    I75 -- ")" --> I300["I300: E → T + id + id ( id id id id id id id id + )·"]
    I75 -- "id" --> I301["I301: E → T + id + id ( id id id id id id id id id·"]
    I76 -- "+" --> I302["I302: E → T + id + id ( id id id id id id id id id + +·"]
    I76 -- "(" --> I303["I303: E → T + id + id ( id id id id id id id id id + (·"]
    I76 -- ")" --> I304["I304: E → T + id + id ( id id id id id id id id id + )·"]
    I76 -- "id" --> I305["I305: E → T + id + id ( id id id id id id id id id id·"]
    I77 -- "+" --> I306["I306: E → T + id + id ( id id id id id id id id id id id + +·"]
    I77 -- "(" --> I307["I307: E → T + id + id ( id id id id id id id id id id id + (·"]
    I77 -- ")" --> I308["I308: E → T + id + id ( id id id id id id id id id id id + )·"]
    I77 -- "id" --> I309["I309: E → T + id + id ( id id id id id id id id id id id id·"]
    I78 -- "+" --> I310["I310: E → T + id + id + id + + +·"]
    I78 -- "(" --> I311["I311: E → T + id + id + id + (·"]
    I78 -- ")" --> I312["I312: E → T + id + id + id + )·"]
    I78 -- "id" --> I313["I313: E → T + id + id + id + id·"]
    I79 -- "+" --> I314["I314: E → T + id + id + id ( + + +·"]
    I79 -- "(" --> I315["I315: E → T + id + id + id ( + + (·"]
    I79 -- ")" --> I316["I316: E → T + id + id + id ( + + )·"]
    I79 -- "id" --> I317["I317: E → T + id + id + id ( + + id·"]
    I80 -- "+" --> I318["I318: E → T + id + id + id ) id + + +·"]
    I80 -- "(" --> I319["I319: E → T + id + id + id ) id + + (·"]
    I80 -- ")" --> I320["I320: E → T + id + id + id ) id + + )·"]
    I80 -- "id" --> I321["I321: E → T + id + id + id ) id + + id·"]
    I81 -- "+" --> I322["I322: E → T + id + id + id id + + +·"]
    I81 -- "(" --> I323["I323: E → T + id + id + id id + (·"]
    I81 -- ")" --> I324["I324: E → T + id + id + id id + )·"]
    I81 -- "id" --> I325["I325: E → T + id + id + id id + id·"]
    I82 -- "+" --> I326["I326: E → T + id + id + id ( id + + +·"]
    I82 -- "(" --> I327["I327: E → T + id + id + id ( id + + (·"]
    I82 -- ")" --> I328["I328: E → T + id + id + id ( id + + )·"]
    I82 -- "id" --> I329["I329: E → T + id + id + id ( id + + id·"]
    I83 -- "+" --> I330["I330: E → T + id + id + id ( ) id + + +·"]
    I83 -- "(" --> I331["I331: E → T + id + id + id ( ) id + + (·"]
    I83 -- ")" --> I332["I332: E → T + id + id + id ( ) id + + )·"]
    I83 -- "id" --> I333["I333: E → T + id + id + id ( ) id + + id·"]
    I84 -- "+" --> I334["I334: E → T + id + id + id ( id id + + +·"]
    I84 -- "(" --> I335["I335: E → T + id + id + id ( id id + + (·"]
    I84 -- ")" --> I336["I336: E → T + id + id + id ( id id + + )·"]
    I84 -- "id" --> I337["I337: E → T + id + id + id ( id id + + id·"]
    I85 -- "+" --> I338["I338: E → T + id + id + id ( id id id + + +·"]
    I85 -- "(" --> I339["I339: E → T + id + id + id ( id id id + + (·"]
    I85 -- ")" --> I340["I339: E → T + id + id + id ( id id id + + )·"]
    I85 -- "id" --> I341["I341: E → T + id + id + id ( id id id + + id·"]
    I86 -- "+" --> I342["I342: E → T + id + id + id ( id id ) id + + +·"]
    I86 -- "(" --> I343["I343: E → T + id + id + id ( id id ) id + + (·"]
    I86 -- ")" --> I344["I344: E → T + id + id + id ( id id ) id + + )·"]
    I86 -- "id" --> I345["I345: E → T + id + id + id ( id id ) id + + id·"]
    I87 -- "+" --> I346["I346: E → T + id + id + id ( id id id id + + +·"]
    I87 -- "(" --> I347["I347: E → T + id + id + id ( id id id id + + (·"]
    I87 -- ")" --> I348["I348: E → T + id + id + id ( id id id id + + )·"]
    I87 -- "id" --> I349["I349: E → T + id + id + id ( id id id id + + id·"]
    I88 -- "+"
```

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Beispiel: Aufbau einer SLR(1)-Syntaxtabelle

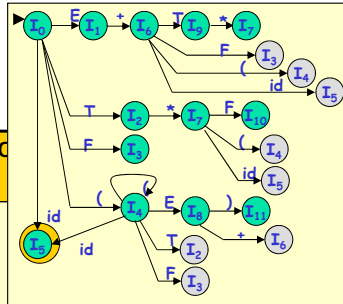
Zust	ACTION						GOTO		
	id	+	*	()	\$	E	T	F
0	s5	-	-	s4	-	-	1	2	3
1	-	s6	-	-	-	acc	-	-	-
2	-	r3	s7	-	r3	r3	-	-	-
3	-	r5	r5	-	r5	r5	-	-	-
4	s5	-	-	s4	-	-	8	2	3
5									
6									
7									
8									
9									
10									
11									



© Prof. J.C. Freytag, Ph.D.

9.61

Zust	ACTION						GOTO		
	id	+	*	()	\$	E	T	F
0	s5	-	-	s4	-	-	1	2	3
1	-	s6	-	-	-	acc	-	-	-
2	-	r3	s7	-	r3	r3	-	-	-
3	-	r5	r5	-	r5	r5	-	-	-
4	s5	-	-	s4	-	-	8	2	3
5		r7	r7	-	r7	r7	-	-	-
6									
7									
8									
9									
10									
11									



I₅
F → id•

Grammatik

- 1) S' → E\$
- 2) E → E + T
- 3) | T
- 4) T → T * F
- 5) | F
- 6) F → (E)
- 7) | id

FOLLOW

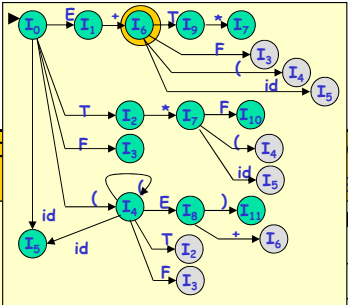
	FOLLOW
E	+,)
T	+, *,)
F	+, *,)

© Prof. J.C. Freytag, Ph.D.

9.62

Vorlesung Compilerbau (SoSe 2018)

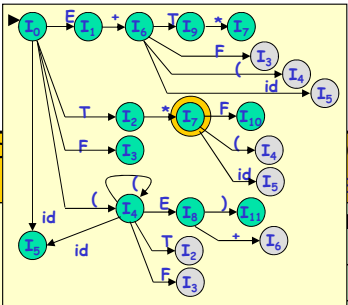
Teil 9: LR/LR(k)-Parsing (2)



The diagram shows LR(0) item sets as nodes and transitions on grammar symbols as edges. The start state is I0. Transitions are labeled with grammar symbols: E, T, F, id, and parentheses. The states are numbered I0 through I10.

Zust	ACTION			GOTO		
	id	+	*		T	F
0	s5	-	-		2	3
1	-	s6	-		-	-
2	-	r3	s7		-	-
3	-	r5	r5	-	-	-
4	s5	-	-	s4	8	2
5	-	r7	r7	-	-	-
6	s5	-	-	s4	-	9
7						
8						
9						
10						
11						

© Prof. J.C. Freytag, Ph.D. 9.63



The diagram shows LR(0) item sets as nodes and transitions on grammar symbols as edges. The start state is I0. Transitions are labeled with grammar symbols: E, T, F, id, and parentheses. The states are numbered I0 through I10.

Zust	ACTION			GOTO		
	id	+	*		T	F
0	s5	-	-		2	3
1	-	s6	-		-	-
2	-	r3	s7		-	-
3	-	r5	r5	-	-	-
4	s5	-	-	s4	8	2
5	-	r7	r7	-	-	-
6	s5	-	-	s4	-	9
7	s5	-	-	s4	-	10
8						
9						
10						
11						

© Prof. J.C. Freytag, Ph.D. 9.64

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

Zust	ACTION			GOTO		
	id	+	*	E	T	F
0	s5	-	-	-	2	3
1	-	s6	-	-	-	-
2	-	r3	s7	-	-	-
3	-	r5	r5	-	-	-
4	s5	-	-	s4	8	2
5	-	r7	r7	-	-	-
6	s5	-	-	s4	-	9
7	s5	-	-	s4	-	10
8	-	s6	-	-	-	-
9	-	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-

Zust	ACTION			GOTO		
	id	+	*	E	T	F
0	s5	-	-	-	2	3
1	-	s6	-	-	-	-
2	-	r3	s7	-	-	-
3	-	r5	r5	-	-	-
4	s5	-	-	s4	8	2
5	-	r7	r7	-	-	-
6	s5	-	-	s4	-	9
7	s5	-	-	s4	-	10
8	-	s6	-	-	-	-
9	-	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-

© Prof. J.C. Freytag, Ph.D. 9.65

Zust	ACTION			GOTO		
	id	+	*	E	T	F
0	s5	-	-	-	2	3
1	-	s6	-	-	-	-
2	-	r3	s7	-	-	-
3	-	r5	r5	-	-	-
4	s5	-	-	s4	8	2
5	-	r7	r7	-	-	-
6	s5	-	-	s4	-	9
7	s5	-	-	s4	-	10
8	-	s6	-	-	-	-
9	-	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-

Zust	ACTION			GOTO		
	id	+	*	E	T	F
0	s5	-	-	-	2	3
1	-	s6	-	-	-	-
2	-	r3	s7	-	-	-
3	-	r5	r5	-	-	-
4	s5	-	-	s4	8	2
5	-	r7	r7	-	-	-
6	s5	-	-	s4	-	9
7	s5	-	-	s4	-	10
8	-	s6	-	-	-	-
9	-	-	-	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-

FOLLOW	
E	+,)
T	+, *,)
F	+, *,)

Grammatik

- 1) $S' \rightarrow E\$$
- 2) $E \rightarrow E + T$
- 3) $E \rightarrow T$
- 4) $T \rightarrow T * F$
- 5) $T \rightarrow F$
- 6) $F \rightarrow (E)$
- 7) $F \rightarrow id$

Actions for I₉:

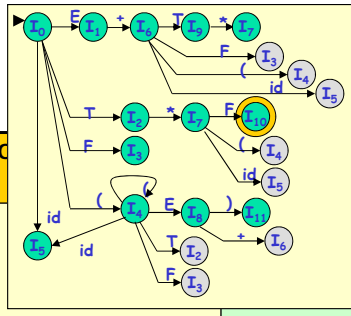
- $E \rightarrow E + T \cdot$ → reduce
- $T \rightarrow T \cdot * F$ → shift

© Prof. J.C. Freytag, Ph.D. 9.66

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)

FOLLOW				ACTION		GOTO	
		id	+	*			
0	s5	-	-	-	-	-	-
1	-	s6	-	-	-	-	-
2	-	r3	s7	-	-	-	-
3	-	r5	r5	-	r5	-	-
4	s5	-	s4	-	-	8	-
5	r7	r7	-	r7	r7	-	-
6	s5	-	s4	-	-	-	9 3
7	s5	-	s4	-	-	-	10
8	-	s6	-	s11	-	-	-
9	-	r2	s7	-	r2	-	-
10	-	r4	r4	-	r4	-	-
11	-	-	-	-	-	-	-



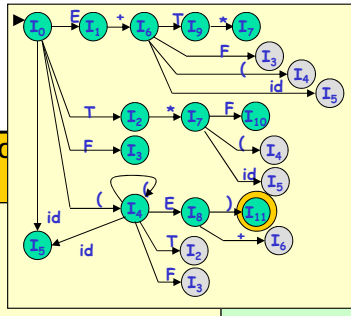
I_{10}
 $T \rightarrow T * F \cdot$

Grammatik

- 1) $S' \rightarrow E \$$
- 2) $E \rightarrow E + T$
- 3) $\quad \mid T$
- 4) $T \rightarrow T * F$
- 5) $\quad \mid F$
- 6) $F \rightarrow (E)$
- 7) $\quad \mid id$

© Prof. J.C. Freytag, Ph.D. 9.67

FOLLOW				ACTION		GOTO	
		id	+	*			
0	s5	-	-	-	-	-	-
1	-	s6	-	-	-	-	-
2	-	r3	s7	-	-	-	-
3	-	r5	r5	-	r5	-	-
4	s5	-	s4	-	-	8	-
5	r7	r7	-	r7	r7	-	-
6	s5	-	s4	-	-	-	9 3
7	s5	-	s4	-	-	-	10
8	-	s6	-	s11	-	-	-
9	-	r2	s7	-	r2	-	-
10	-	r4	r4	-	r4	-	-
11	-	r6	r6	-	r6	-	-



I_{11}
 $F \rightarrow (E) \cdot$

Grammatik

- 1) $S' \rightarrow E \$$
- 2) $E \rightarrow E + T$
- 3) $\quad \mid T$
- 4) $T \rightarrow T * F$
- 5) $\quad \mid F$
- 6) $F \rightarrow (E)$
- 7) $\quad \mid id$

© Prof. J.C. Freytag, Ph.D. 9.68

SLR(1)-Grammatik

- Jede SLR(1)-Grammatik ist eindeutig
- aber nicht jede eindeutige Grammatik ist vom SLR(1)-Typ

■ Beispiel

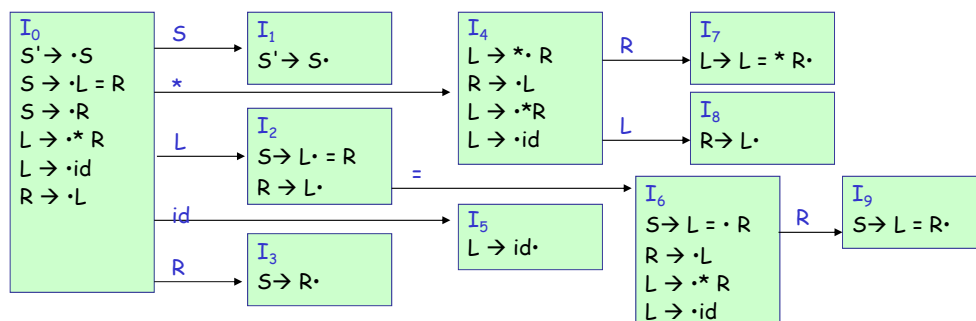
Grammatik

- 1) $S \rightarrow L = R$
- 2) $S \rightarrow R$
- 3) $L \rightarrow *R$
- 4) $L \rightarrow \text{id}$
- 5) $R \rightarrow L$

© Prof. J.C. Freytag, Ph.D.

9.69

Eindeutige Grammatik mit Shift-Reduce-Konflikt

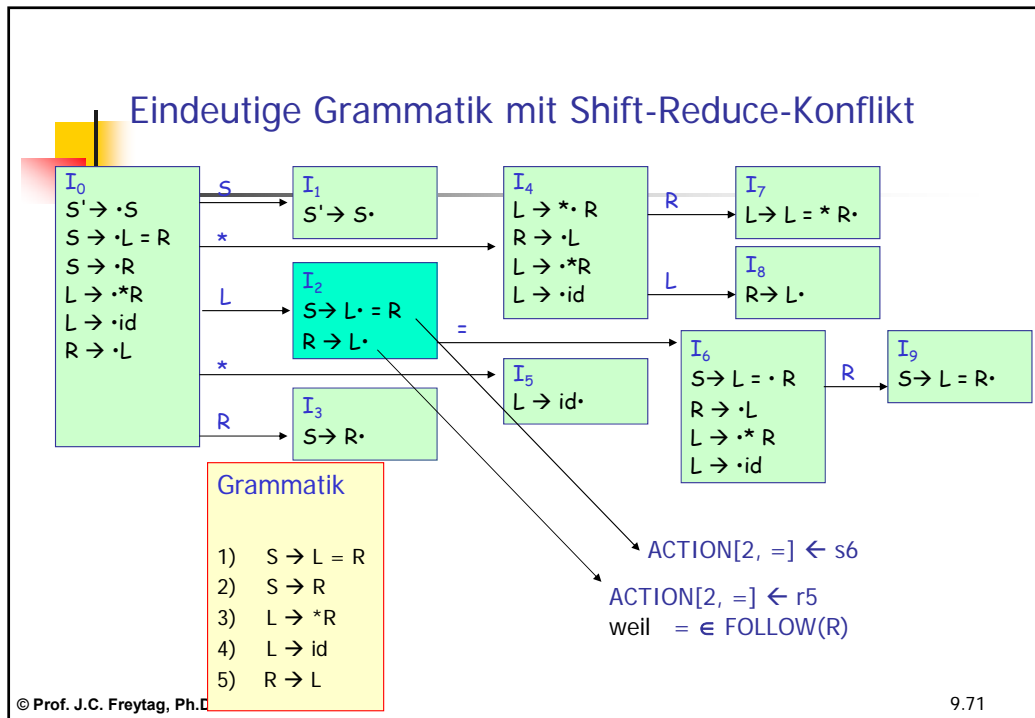


© Prof. J.C. Freytag, Ph.D.

9.70

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



Zwischenfazit

Grammatik

- 1) $S \rightarrow L = R$
- 2) $S \rightarrow R$
- 3) $L \rightarrow * R$
- 4) $L \rightarrow id$
- 5) $R \rightarrow L$

- obwohl Grammatik eindeutig, ist sie nicht von einem SLR-Parser analysierbar
- Grammatik ist nicht vom Typ SLR(1)
- ➔ SLR-Parser sind nicht mächtig genug

drei Techniken zur **Konstruktion** von LR-Syntaxanalysetabellen

- (1) einfache LR- (kurz: SLR-) Analyse
- (2) **kanonische LR-Analyse**
- (3) vorrausschauende LR- (kurz: LALR-) Analyse

© Prof. J.C. Freytag, Ph.D. 9.72

Vorlesung Compilerbau (SoSe 2018)

Teil 9: LR/LR(k)-Parsing (2)



Fragen???

