



# Vorlesung Compilerbau (SoSe2018)


## Teil 6: Parsing III - LR-Parsing (1)



Vorlesung Compilerbau:  
Syntaktische Analyse – Parsing III  
LR-Parsing (1)

Vorlesung des BA-Studiums  
Prof. Johann Christoph Freytag, Ph.D.  
Institut für Informatik, Humboldt-Universität zu Berlin  
SoSe2018

© Prof. J.C. Freytag, Ph.D. 6.1



---

Kellerautomat (Kurzeinführung)

© Prof. J.C. Freytag, Ph.D. 6.2



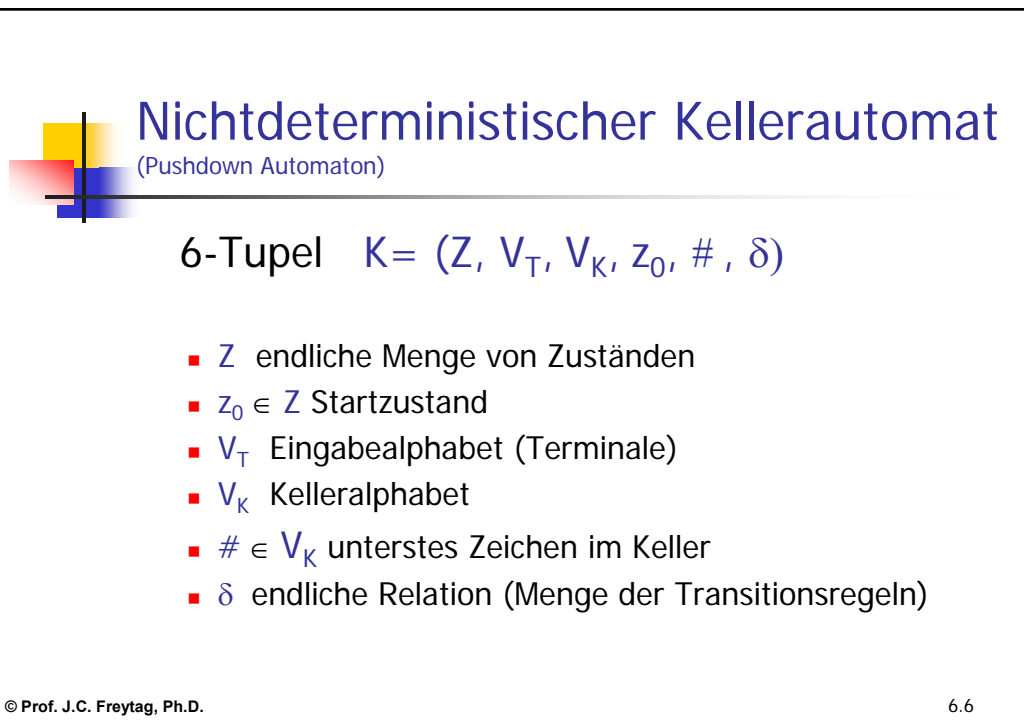
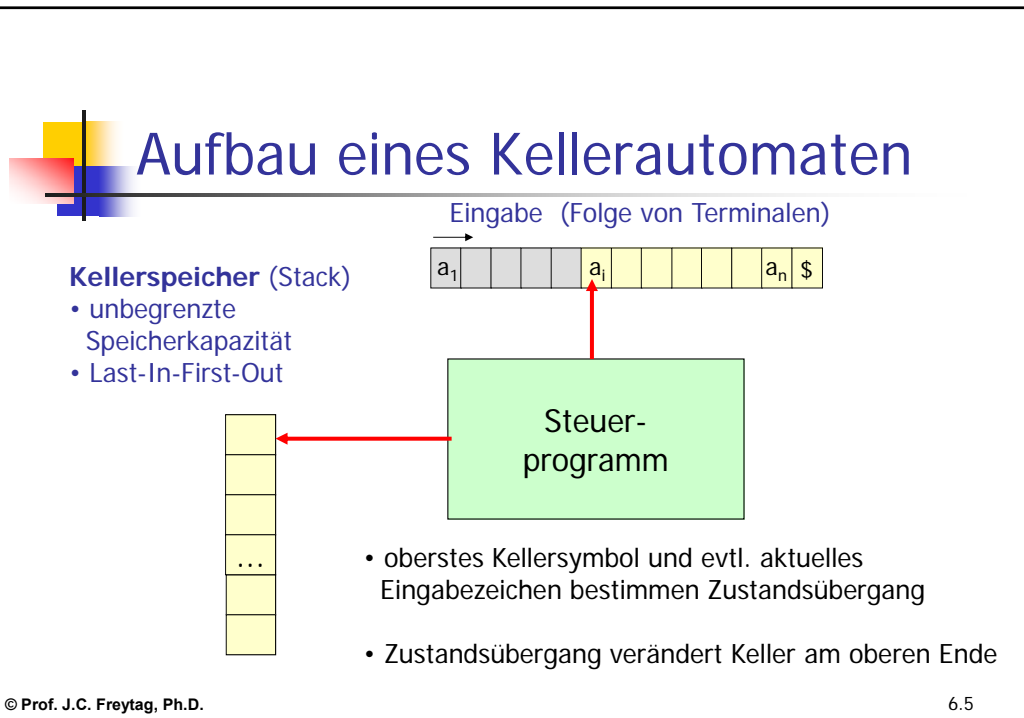
## Theoretische Grundlagen

- Äquivalenz  
kontextfreier Grammatiken und Kellerautomaten
  - Zu jeder kontextfreien Grammatik kann man einen Kellerautomaten konstruieren, der die von der Grammatik definierte Sprache akzeptiert.
  - Die von einem Kellerautomaten akzeptierte Sprache ist kontextfrei.



## Gedächtnis des Automaten

- Welche Art von »Gedächtnis« ist für kontextfreie Sprachen notwendig?
  - $L = \{a^n b c^n \mid n > 0\}$
  - man muss sich Anzahl der  $a$  merken
- in einem (Keller-)Speicher sollten
  - die  $a$  gespeichert werden können und
  - nachdem ein  $b$  erkannt wurde, sollten mit den erkannten  $c$  die  $a$  wieder aus dem Keller entfernt werden können



## Zustandsübergang (NDKA)

- Zustandsübergangstransition (*state transition*)

$$\delta: Z \times (V_t \cup \{\varepsilon\}) \times V_k \rightarrow (Z \times V_k^*)$$

- befindet sich Automat  $K$ 
  - im Zustand  $z$  und
  - liest das Eingabezeichen  $t$  und
  - sei  $A$  das oberste Kellerelement, so kann  $K$  im nächsten Schritt
  - in den Zustand  $z'$  übergehen und
  - das oberste Kellerelement durch eine Symbolfolge  $B_1 \dots B_k$  ersetzen ( $B_1$  ist dabei das oberste Kellerelement)

### Spezialfall

- $\varepsilon$ -Eingabe: spontaner Übergang (ohne Konsumtion eines Eingabe-Signals)

© Prof. J.C. Freytag, Ph.D.

6.7

## Akzeptanz, Startzustände

- Akzeptanz eines Eingabewortes
  - **keine Endzustände**
  - Akzeptanz erfolgt,
    - wenn der Keller leer ist, i.e. nur noch das Zeichen  $\#$  zu finden ist (zu Beginn jeder Analyse steht das Zeichen  $\#$  im Keller)
    - ... **und** die Eingabe leer (i.e. das  $\$$ -Zeichen gelesen wurde).
- Startzustände
  - Einschränkung auf **einen** Startzustand ist nicht zwingend
    - mittels "spontaner" Übergänge kann in die "eigentlichen Startzustände" übergegangen werden

© Prof. J.C. Freytag, Ph.D.

6.8

### Konfiguration

- eine **Konfiguration**  $k$  ist ein Tripel  $(z, w, v) \in (Z \times V_T^* \times V_K^*)$  mit
  - $z$  : aktueller Zustand  $z$
  - $w$  : (noch zu lesende) Eingabewort
  - $v$  : Kellerzustand  $v$

Eine **Konfiguration** beschreibt eine "**Momentaufnahme**" des Kellerautomaten

- Zustand,
  - noch zu lesendes Eingabewort,
  - Kellerzustand (oberstes Kellerzeichen links)
- Keller: *nur das oberste Zeichen des Kellers kann gelesen werden*

### „Rechenschritt“, akzeptierte Sprache

- zweistellige Relation  $\Rightarrow$ , die eine Konfiguration in die Nachfolgekongfiguration überführt:

$k \Rightarrow k'$  ( $k'$  ergibt sich durch *einmalige* Anwendung der  $\delta$ -Funktion)

- sei  $\Rightarrow^*$  die reflexive, transitive Hülle von  $\Rightarrow$ , dann ist die durch einen Kellerautomaten  $K$  akzeptierte Sprache  $L(K)$  folgendermaßen bestimmt:

$L(K) = \{w \in V_T^* \mid (z_0, w, \#) \Rightarrow^* (z, \varepsilon, \#) \text{ für ein } z \in Z\}$

(**Achtung:** „ $\Rightarrow$ “ neu definiert im Kontext des Kellerautomaten)

#### Satz:

Eine Sprache  $L$  ist **kontextfrei** gdw. (iff)  $L$  von einem nichtdeterministischen Kellerautomaten erkannt wird.

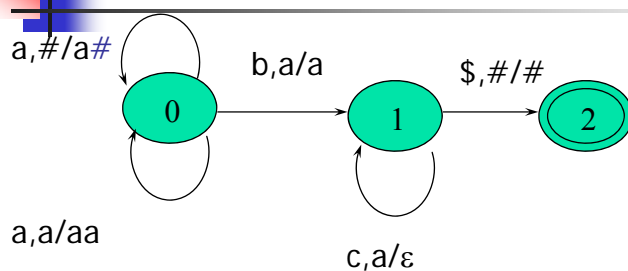
## Deterministischer Kellerautomat

- Deterministischer Kellerautomat
  - **Spezialfall** des Nichtdeterministischen Kellerautomaten
  - darf nur eine Transition pro (Eingabe, Keller)-Paar besitzen
- Deterministischer Kellerautomat akzeptiert per Endzustand und nicht per leerem Keller
- **Bemerkung:**  
bei Nichtdeterministischen Kellerautomaten sind beide Akzeptanzmechanismen (leerer Keller bzw. Endzustand) äquivalent

© Prof. J.C. Freytag, Ph.D.

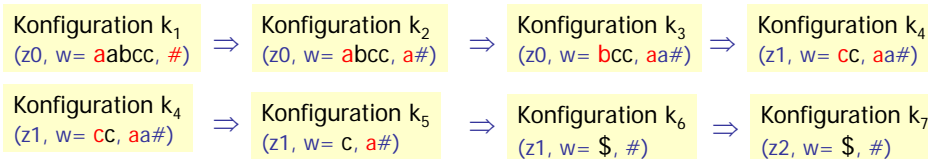
6.11

## Beispiel: $L = \{a^nbc^n \mid n > 0\}$



### Beispiel- Eingaben

1. aabcc
2. abcc




© Prof. J.C. Freytag, Ph.D.

6.12

# Vorlesung Compilerbau (SoSe2018)

## Teil 6: Parsing III - LR-Parsing (1)


 Transitionstabelle

$L = \{xcx^R \mid x \in \{a,b\}^*\}$

Reverse

Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$

© Prof. J.C. Freytag, 6.13

 Transitionstabelle

$L = \{xcx^R \mid x \in \{a,b\}^*\}$


Reverse

Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$

© Prof. J.C. Freytag, 6.14

# Vorlesung Compilerbau (SoSe2018)

## Teil 6: Parsing III - LR-Parsing (1)




### Transitionstabelle

Reverse

$$L = \{xcx^R \mid x \in \{a,b\}^*\}$$

Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a a)$

© Prof. J.C. Freytag,
6.15



### Transitionstabelle

Reverse

$$L = \{xcx^R \mid x \in \{a,b\}^*\}$$


Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a a)$
4	$z_0$	b	a	$(z_0, b a)$

© Prof. J.C. Freytag,
6.16



# Vorlesung Compilerbau (SoSe2018)

## Teil 6: Parsing III - LR-Parsing (1)




### Transitionstabelle

Reverse

$$L = \{xcx^R \mid x \in \{a,b\}^*\}$$

Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a a)$
4	$z_0$	b	a	$(z_0, b a)$
5	$z_0$	a	b	$(z_0, a b)$

© Prof. J.C. Freytag,
6.17



### Transitionstabelle

Reverse


$$L = \{xcx^R \mid x \in \{a,b\}^*\}$$

Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a a)$
4	$z_0$	b	a	$(z_0, b a)$
5	$z_0$	a	b	$(z_0, a b)$
6	$z_0$	b	b	$(z_0, b b)$

© Prof. J.C. Freytag,
6.18

# Vorlesung Compilerbau (SoSe2018)

## Teil 6: Parsing III - LR-Parsing (1)




### Transitionstabelle

$$L = \{xcx^R \mid x \in \{a,b\}^*\}$$

Reverse

Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a \ a)$
4	$z_0$	b	a	$(z_0, b \ a)$
5	$z_0$	a	b	$(z_0, a \ b)$
6	$z_0$	b	b	$(z_0, b \ b)$
7	$z_0$	c	#	$(z_1, \#)$

© Prof. J.C. Freytag, \_\_\_\_\_
 6.19



### Transitionstabelle

$$L = \{xcx^R \mid x \in \{a,b\}^*\}$$


Reverse

Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a \ a)$
4	$z_0$	b	a	$(z_0, b \ a)$
5	$z_0$	a	b	$(z_0, a \ b)$
6	$z_0$	b	b	$(z_0, b \ b)$
7	$z_0$	c	#	$(z_1, \#)$
8	$z_0$	c	a	$(z_1, a)$

© Prof. J.C. Freytag, \_\_\_\_\_
 6.20

# Vorlesung Compilerbau (SoSe2018)

## Teil 6: Parsing III - LR-Parsing (1)




### Transitionstabelle

Reverse

$$L = \{xcx^R \mid x \in \{a, b\}^*\}$$

Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a \ a)$
4	$z_0$	b	a	$(z_0, b \ a)$
5	$z_0$	a	b	$(z_0, a \ b)$
6	$z_0$	b	b	$(z_0, b \ b)$
7	$z_0$	c	#	$(z_1, \#)$
8	$z_0$	c	a	$(z_1, \ a)$
9	$z_0$	c	b	$(z_1, \ b)$

© Prof. J.C. Freytag,
6.21



### Transitionstabelle

Reverse

$$L = \{xcx^R \mid x \in \{a, b\}^*\}$$

Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \ #)$
3	$z_0$	a	a	$(z_0, a \ a)$
4	$z_0$	b	a	$(z_0, b \ a)$
5	$z_0$	a	b	$(z_0, a \ b)$
6	$z_0$	b	b	$(z_0, b \ b)$
7	$z_0$	c	#	$(z_1, \ #)$
8	$z_0$	c	a	$(z_1, \ a)$
9	$z_0$	c	b	$(z_1, \ b)$
10	$z_1$	a	a	$(z_1, \ \epsilon)$

© Prof. J.C. Freytag,
6.22

# Vorlesung Compilerbau (SoSe2018)

## Teil 6: Parsing III - LR-Parsing (1)

### Transitionstabelle

$L = \{xcx^R \mid x \in \{a,b\}^*\}$

Reverse

Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a \ a)$
4	$z_0$	b	a	$(z_0, b \ a)$
5	$z_0$	a	b	$(z_0, a \ b)$
6	$z_0$	b	b	$(z_0, b \ b)$
7	$z_0$	c	#	$(z_1, \#)$
8	$z_0$	c	a	$(z_1, a)$
9	$z_0$	c	b	$(z_1, b)$
10	$z_1$	a	a	$(z_1, \varepsilon)$
11	$z_1$	b	b	$(z_1, \varepsilon)$

© Prof. J.C. Freytag, 6.23

### Beispiel: $L = \{xcx^R \mid x \in \{a,b\}^*\}$

```

graph LR
    0((0)) -- "a, #/a#  
b, #/b#" --> 0
    0 -- "c, a/a  
c, b/b  
c, #/#" --> 1((1))
    1 -- "a, a/ε  
b, b/ε" --> 1
    1 -- "$, #/#" --> 2(((2)))
  
```

© Prof. J.C. Freytag, Ph.D. 6.24

# Vorlesung Compilerbau (SoSe2018)

## Teil 6: Parsing III - LR-Parsing (1)

### Transitionstabelle

$L = \{xcx^R \mid x \in \{a, b\}^*\}$

Reverse

Index	Zustand	Eing.	Keller	Transition
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a a)$
4	$z_0$	b	a	$(z_0, b a)$
5	$z_0$	a	b	$(z_0, a b)$
6	$z_0$	b	b	$(z_0, b b)$
7	$z_0$	c	#	$(z_1, \#)$
8	$z_0$	c	a	$(z_1, a)$
9	$z_0$	c	b	$(z_1, b)$
10	$z_1$	a	a	$(z_1, \varepsilon)$
11	$z_1$	b	b	$(z_1, \varepsilon)$
12	$z_1$	\$	#	$(z_2, \#)$

© Prof. J.C. Freytag, 6.25

### Nicht-Determinismus

$L = \{xx^R \mid x \in \{a, b\}^*\}$

Index	Zustand	Eingabe	Keller	Transit.
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a a)$
4	$z_0$	b	a	$(z_0, b a)$
5	$z_0$	a	b	$(z_0, a b)$
6	$z_0$	b	b	$(z_0, b b)$

© Prof. J.C. Freytag, 6.26

# Vorlesung Compilerbau (SoSe2018)

## Teil 6: Parsing III - LR-Parsing (1)

### Nicht-Determinismus

$$L = \{xx^R \mid x \in \{a, b\}^*\}$$

Index	Zustand	Eingabe	Keller	Transit.
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a a)$
4	$z_0$	b	a	$(z_0, b a)$
5	$z_0$	a	b	$(z_0, a b)$
6	$z_0$	b	b	$(z_0, b b)$
7	$z_0$	$\epsilon$	#	$(z_1, \#)$
8	$z_0$	$\epsilon$	a	$(z_1, a)$
9	$z_0$	$\epsilon$	b	$(z_1, b)$

© Prof. J.C. Freytag,

6.27

### Nicht-Determinismus

$$L = \{xx^R \mid x \in \{a, b\}^*\}$$

Index	Zustand	Eingabe	Keller	Transit.
1	$z_0$	a	#	$(z_0, a \#)$
2	$z_0$	b	#	$(z_0, b \#)$
3	$z_0$	a	a	$(z_0, a a)$
4	$z_0$	b	a	$(z_0, b a)$
5	$z_0$	a	b	$(z_0, a b)$
6	$z_0$	b	b	$(z_0, b b)$
7	$z_0$	$\epsilon$	#	$(z_1, \#)$
8	$z_0$	$\epsilon$	a	$(z_1, a)$
9	$z_0$	$\epsilon$	b	$(z_1, b)$
10	$z_1$	a	a	$(z_1, \epsilon)$
11	$z_1$	b	b	$(z_1, \epsilon)$
12	$z_1$	\$	#	$(z_2, \#)$

© Prof. J.C. Freytag, P

6.28

### Nicht-Determinismus

- Nicht-Determinismus:
  - es existieren **mehrere** Transitionen für die **gleiche** Eingabe-Keller Konfiguration
  - d.h. aus demselben Zustand heraus existiert für den Keller=Y und die Eingabe=w, eine **weitere andere** Transition mit Keller=Y Eingabe=w

© Prof. J.C. Freytag, Ph.D.

6.29

### Nichtdeterminismus - Determinismus

- **einige** nicht-deterministische Kellerautomaten **können** in **deterministische** umgewandelt werden
- andere kontextfreie Sprachen sind dagegen **inhärent** nicht-deterministisch

$$L = \{x \in \{a,b\}^* \mid x = x^R\}$$

#### Palindrome (von Hansgeorg Stengel)

- |               |                      |                            |
|---------------|----------------------|----------------------------|
| • OTTO        | • LAGERREGAL         | • ALLE BANANEN ANABELLA    |
| • ANNASUSANNA | • RELIEFPFEILER      | • REGINE WEINE NIE WENIGER |
| • NURDUGUDRUN | • DIENSTMANNAMTSNEID | • EIN MAKI KAM NIE         |

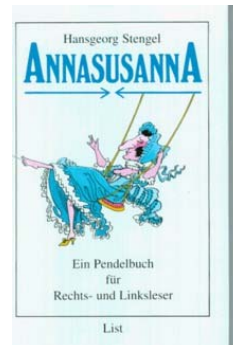
© Prof. J.C. Freytag, Ph.D.

6.30

## noch ein Stengel-Palindrom

LEONI LEG ANTONS NOTNAGEL IN OEL

**Hansgeorg Stengel:**  
»Annasusanna – Ein Pendelbuch für  
Rechts- und Linksleser«  
Eulenspiegel Verlag Berlin, 1984



© Prof. J.C. Freytag, Ph.D.

6.31

## Anwendung im Compilerbau

- Ausgang
  - beliebige kontextfreie Grammatik
- Konstruktion
  - eines Kellerautomaten, der die durch die Grammatik definierte Sprache akzeptiert
- Problem
  - Kellerautomat ist nichtdeterministisch (auch wenn die Ausgangsgrammatik einen deterministischen Akzeptor haben sollte)
  - Aber: für gewisse Unterklassen Kontextfreier Grammatiken können jedoch deterministische Kellerautomaten abgeleitet werden.

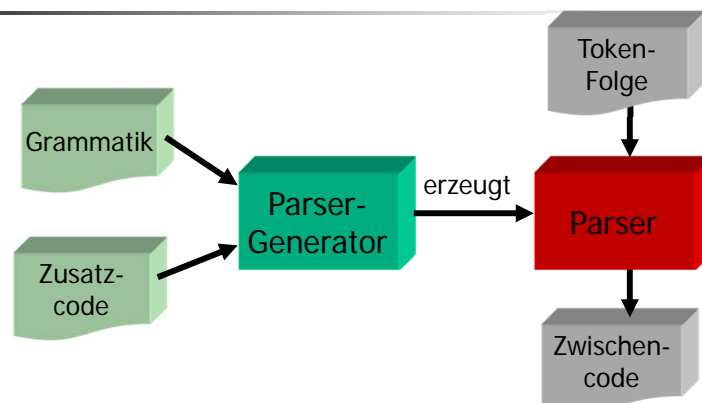
© Prof. J.C. Freytag, Ph.D.

6.32



... zurück zum Parsen

## Konstruktion eines LR-Parsers



**Ziel:** Automatische Konstruktion eines flexiblen Parsers



### LR(k)- Analyseverfahren

- LR-Verfahren
  - **L**: Links-nach-rechts-Verarbeitung der Eingabefile
  - **R**: Rechtsableitung in umgekehrter Reihenfolge (Bottom-Up-Verfahren)
  - **k**: Zahl der voraus betrachteten Eingabesymbole (*prädiktives* Verfahren)
  - ohne Rücksetzen und damit effizient (lineare Komplexität)
- LR-Grammatiken

© Prof. J.C. Freytag, Ph.D.

6.35



### Attraktivität von LR- Verfahren

- können praktisch alle Programmiersprachenkonstrukte erkennen, für die kontextfreie Grammatiken existieren
- LR ist die *allgemeinste Shift-Reduce-Syntaxanalyse ohne Backtracking*;
  - dennoch kann sie effizienter implementiert werden als andere Shift-Reduce-Methoden
- LR-Sprachen enthalten LL-Sprachen!!
- **Nachteil**: sehr großer Aufwand für Hand-Implementation

aber: automatisch (mittels yacc/bison) erzeugter Parser erkennen und lokalisieren Mehrdeutigkeiten oder andere schwierige syntaktische Konstrukte kontextfreier Grammatiken

© Prof. J.C. Freytag, Ph.D.

6.36



## Bottom-Up-Syntaxanalyseverfahren

- einfaches Verfahren:  
Shift-Reduce-Syntaxanalyse
  - davon einfach zu implementieren:  
Operator-Precedence-Syntaxanalyse
- allgemeineres Verfahren:  
LR-Syntaxanalyse



## Prinzip der Bottom-Up-Verfahren

- sukzessive Anwendung von Reduktionsschritten, um letztendlich ein Eingabewort  $w$  (Programm) auf das Startsymbol  $S$  der Grammatik zu reduzieren
- einzelner (Reduktions-) Schritt:

Teilzeichenkette	→	Symbol
<i>die mit rechter Regel-Seite übereinstimmt</i>		<i>das mit der linken Regel-Seite übereinstimmt</i>
- wird die Teilzeichenkette *richtig* gewählt, liegt eine Rechtsableitung in umgekehrter Reihenfolge vor

# Vorlesung Compilerbau (SoSe2018)

## Teil 6: Parsing III - LR-Parsing (1)

### Beispiel

Kandidaten (aus FIRST-Menge für A und B)

**Grammatik:**

$$\begin{aligned} S &\rightarrow aABe \\ A &\rightarrow Abc \mid b \\ B &\rightarrow d \end{aligned}$$

**Eingabewort**

abbcde

Beobachtung

tatsächlich geben diese 4 Reduktionen in umgekehrter Reihenfolge folgende Rechtsableitung wieder:

$$S \rightarrow aABe \rightarrow aAde \rightarrow aAbcde \rightarrow abbcde$$

© Prof. J.C. Freytag, Ph.D. 6.39


### Prinzip der Bottom-Up-Verfahren(2)

- Ziel:  
bei gegebenem Eingabewort  $w$  und einer Grammatik  $G$  wird ein Parsebaum generiert, dessen Konstruktion an den Blättern beginnt und in Richtung Wurzel wächst

© Prof. J.C. Freytag, Ph.D. 6.40

# Vorlesung Compilerbau (SoSe2018)

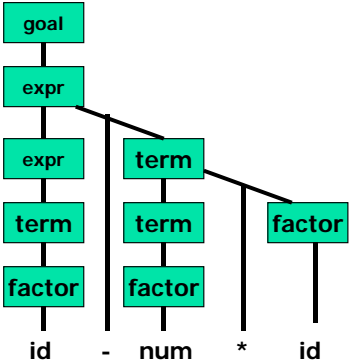
## Teil 6: Parsing III - LR-Parsing (1)




### Beispiel

**Grammatik:**

1.  $\langle \text{goal} \rangle ::= \langle \text{expr} \rangle$
2.  $\langle \text{expr} \rangle ::= \langle \text{expr} \rangle + \langle \text{term} \rangle$
3.  $\quad \quad \quad | \langle \text{expr} \rangle - \langle \text{term} \rangle$
4.  $\quad \quad \quad | \langle \text{term} \rangle$
5.  $\langle \text{term} \rangle ::= \langle \text{term} \rangle * \langle \text{factor} \rangle$
6.  $\quad \quad \quad | \langle \text{term} \rangle / \langle \text{factor} \rangle$
7.  $\quad \quad \quad | \langle \text{factor} \rangle$
8.  $\langle \text{factor} \rangle ::= \text{num}$
9.  $\quad \quad \quad | \text{id}$



© Prof. J.C. Freytag, Ph.D. 6.41



### Beispiel

- Grammatik:
  - 1  $S \rightarrow aABe$
  - 2  $A \rightarrow Abc$
  - 3  $\quad \quad | b$
  - 4  $B \rightarrow d$
- ... und die Eingabe: abbcde

Prodkt.	Satzform
3	a <span style="background-color: #00FF00;">b</span> bcde
2	a <span style="background-color: #00FF00;">A</span> bcde
4	a A <span style="background-color: #00FF00;">d</span> e
1	<span style="background-color: #00FF00;">aABe</span>
-	S

- Trick: Finden der richtigen (ersten – linken) Satzform

© Prof. J.C. Freytag, Ph.D. 6.42

# Vorlesung Compilerbau (SoSe2018)

## Teil 6: Parsing III - LR-Parsing (1)

### »Handle«

#### Ziel

- Suche nach einem Teilstring  $\alpha$ , der mit der rechten Seite einer Regel  $A \rightarrow \alpha$  übereinstimmt und dessen Reduktion zum Nichtterminalen  $A$  in einer umgekehrten Rechtsableitung reduziert werden kann
- der Teilstring  $\alpha$  wird auch **Griff** (engl. »Handle«) genannt

#### formale Definition

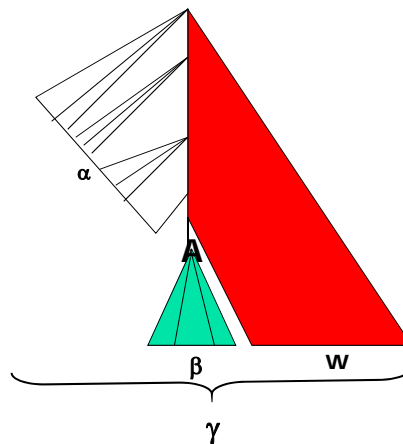
- Ein »Handle« einer Zeichenkette  $\gamma$  (die aus Terminalen und Nichtterminalen bestehen kann und einer Rechtsableitung entstammt) ist
  - eine Regel  $A \rightarrow \beta$  und
  - eine Position in  $\gamma$ , in der  $\beta$  gefunden und durch  $A$  ersetzt wird

© Prof. J.C. Freytag, Ph.D.

6.43

### »Handles« (1)

Der »Handle«  $A \rightarrow \beta$  im Parsebaum für rechtsabgeleiteten String  $\gamma = \alpha\beta w$



© Prof. J.C. Freytag, Ph.D.

6.44

### »Handle« (2)

#### Bemerkung

- falls  $s \Rightarrow^* \alpha A w \Rightarrow \alpha \beta w$ , dann ist  $A \rightarrow \beta$  an der Position, die  $\alpha$  folgt, ein »Handle« für  $\gamma = \alpha \beta w$
- da  $\gamma$  einer Rechtsableitung entstammt, enthält das Teilwort  $w$  **rechts vom »Handle« nur Terminalsymbole** (deswegen  $w \in V_t^*$ ).

#### Beobachtung für letztes Beispiel

$S \Rightarrow aABe \Rightarrow aAde \Rightarrow aAbcde \Rightarrow abbcde$

#### Grammatik:

$S \rightarrow aABe$   
 $A \rightarrow Abc \mid b$   
 $B \rightarrow d$

$abbcde$  entstammt einer Rechtsableitung, Handle:  $A \rightarrow b$  an Position 2

$aAbcde$  entstammt einer Rechtsableitung, Handle:  $A \rightarrow Abc$  an Position 2

Kurzsprechweise: Teilstring  $Abc$  ist Handle von  $aAbcde$

© Prof. J.C. Freytag, Ph.D.

6.45

### »Handle« (3)

#### Satz:

- Falls die Grammatik  $G$  eindeutig ist, dann gibt es in jedem String einer Rechtsableitung ein eindeutig bestimmte »Handle«

Beweisidee (durch Anwendung der Definitionen):

- $G$  ist eindeutig  $\Rightarrow$  Rechtsableitung ist eindeutig
  - $\Rightarrow$  es existiert eine eindeutige Regel  $A \rightarrow \beta$ , die  $\gamma_{i-1}$  nach  $\gamma_i$  überführt
  - $\Rightarrow$  es existiert eine eindeutige Position  $k$ , an der die Regel  $A \rightarrow \beta$  angewandt wird
  - $\Rightarrow$  dies ist das eindeutige »Handle«  $A \rightarrow \beta$

© Prof. J.C. Freytag, Ph.D.

6.46

# Vorlesung Compilerbau (SoSe2018)

## Teil 6: Parsing III - LR-Parsing (1)

### Beispiel "Handle"

1.  $\langle \text{goal} \rangle ::= \langle \text{expr} \rangle$
2.  $\langle \text{expr} \rangle ::= \langle \text{expr} \rangle + \langle \text{term} \rangle$
3.       |  $\langle \text{expr} \rangle - \langle \text{term} \rangle$
4.       |  $\langle \text{term} \rangle$
5.  $\langle \text{term} \rangle ::= \langle \text{term} \rangle * \langle \text{factor} \rangle$
6.       |  $\langle \text{term} \rangle / \langle \text{factor} \rangle$
7.       |  $\langle \text{factor} \rangle$
8.  $\langle \text{factor} \rangle ::= \text{num}$
9.       |  $\text{id}$

Prodkt.	Satzform
-	$\langle \text{goal} \rangle$
1	$\langle \text{expr} \rangle$
3	$\langle \text{expr} \rangle - \langle \text{term} \rangle$
5	$\langle \text{expr} \rangle - \langle \text{term} \rangle * \langle \text{factor} \rangle$
9	$\langle \text{expr} \rangle - \langle \text{term} \rangle * \text{id}$
7	$\langle \text{expr} \rangle - \langle \text{factor} \rangle * \text{id}$
8	$\langle \text{expr} \rangle - \text{num} * \text{id}$
4	$\langle \text{term} \rangle - \text{num} * \text{id}$
7	$\langle \text{factor} \rangle - \text{num} * \text{id}$
9	$\text{id} - \text{num} * \text{id}$

© Prof. J.C. Freytag, Ph.D.

6.47

### »Handle-Pruning«

**Def.:** Prozess, einen Parsebaum »bottom-up« zu konstruieren, heißt »**handle-pruning**«

- sei  $w$  String einer Rechtsableitung  
 $S \Rightarrow \gamma_0 \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \dots \Rightarrow \gamma_{n-1} \Rightarrow \gamma_n = w$
- dann ist folgender einfacher Algorithmus ein (Parse-) handle-pruning-Algorithmus  
**for**  $i = n$  **downto**  $0$  **do**  
     finde die Handle  $A_i \rightarrow \beta_i$  in  $\gamma_i$  ;  
     ersetze  $\beta_i$  durch  $A_i$ , um  $\gamma_{i-1}$  zu generieren  
**endfor**

© Prof. J.C. Freytag, Ph.D.

6.48





## Shift-Reduce-Parser

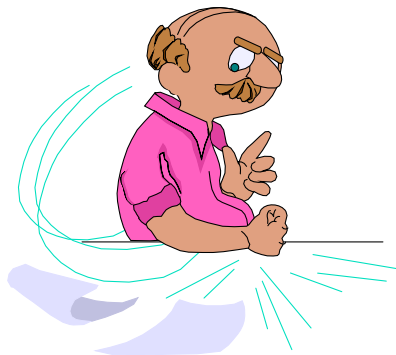
- Nächster Foliensatz

© Prof. J.C. Freytag, Ph.D.

7.49



## Fragen???



© Prof. J.C. Freytag, Ph.D.

6.50