

Die Lehrveranstaltung wird heute aufgezeichnet. Sie sind nicht verpflichtet, Ihr Videobild, Ihren Audioton oder andere personenbezogene Daten wie einen Klarnamen oder ein Bild von Ihnen zu übertragen. Die Übertragung von Video und Audio ist für Sie aktuell deaktiviert. Wenn Sie nicht Teil der Aufzeichnung werden möchten, aktivieren Sie die Übertragung einfach nicht. Wenn Sie hingegen einverstanden sind, Teil der Aufzeichnung zu werden, lesen Sie bitte vorher die Einwilligungserklärung und die Informationen zur Verarbeitung Ihrer personenbezogenen Daten sorgfältig durch. Diese finden Sie ganz unten im Moodle-Lernraum verlinkt. Wenn Sie dann Ihr Bild, Ihr Video und/oder Ihr Audio einschalten, willigen Sie in deren Aufzeichnung gemäß Einwilligungserklärung ein.

Kapitel 9

Algorithmen für kontextfreie
Sprachen

Abschnitt 9.1

Parsing

- ▶ Die Syntax von Programmiersprachen wie JAVA ist typischerweise durch kontextfreie Grammatiken spezifiziert.
- ▶ Ein Programm wird dem Compiler einfach als Textfile, oder abstrakter, als Wort über dem Terminalalphabet, gegeben.
- ▶ Der Compiler muss zunächst die Struktur des Programms rekonstruieren; dies geschieht, indem er einen Ableitungsbaum für das Programm konstruiert.
Diesen Prozess nennt man **Parsing** und den Teil des Compilers, der ihn ausführt, **Parser**.
- ▶ Parser werden aber nicht nur in Compilern benötigt, sondern auch anderen Stellen, wo man strukturierte Information aus einem Textfile rekonstruieren muss.
Beispielsweise benötigt ein Webbrowser einen HTML-Parser.

Wir betrachten die Sprache L_T der arithmetischen Terme mit den Operatoren $+$ und $*$ (zweistellig) und $-$ (einstellig), den Konstanten 0 und 1 und Variablen x, y, z . Das Alphabet ist

$$\Sigma_T := \{+, *, -, 0, 1, x, y, z, (,)\}.$$

Folgende kontextfreie Grammatik \mathcal{G}_T erzeugt die Terme:

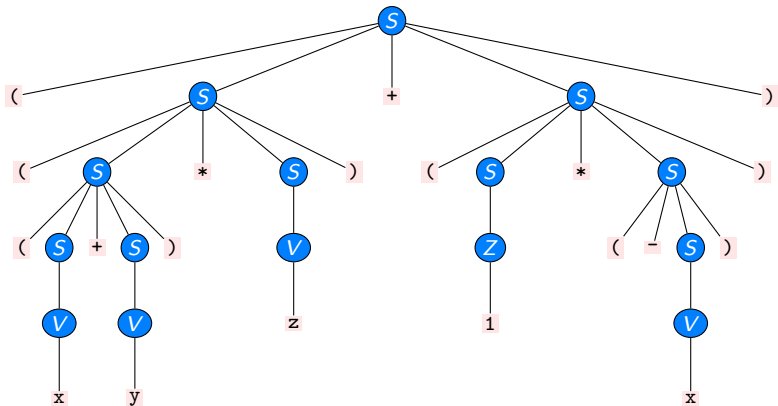
$$S \rightarrow (S+S) \mid (S*S) \mid (-S) \mid Z \mid V$$

$$Z \rightarrow 0 \mid 1$$

$$V \rightarrow x \mid y \mid z$$

Beispiel 9.1 (Forts.)

Ableitungsbaum für das Wort $((x+y)*z)+(1*(-x)) \in L_T$:



Die Grammatik \mathcal{G}_T ist **eindeutig**; das heißt, jedes Wort besitzt einen eindeutigen Ableitungsbaum.

Ein Parser für die Grammatik \mathcal{G}_T

PARSE(*text*)

- 1: $a \leftarrow$ first symbol of *text*
- 2: initialise stack \mathcal{S}
- 3: **while** $a \neq$ end of file **do**
- 4: **if** $a = ($ or $a = +$ or $a = *$ or $a = -$ **then**
- 5: PUSH(\mathcal{S}, a)

Ein Parser für die Grammatik \mathcal{G}_T

PARSE(*text*)

```
1:  $a \leftarrow$  first symbol of text
2: initialise stack  $\mathcal{S}$ 
3: while  $a \neq$  end of file do
4:   if  $a = ($  or  $a = +$  or  $a = *$  or  $a = -$  then
5:     PUSH( $\mathcal{S}$ ,  $a$ )
6:   else if  $a = 0$  or  $a = 1$  then
7:      $T \leftarrow$  NUMTREE( $a$ ); PUSH( $\mathcal{S}$ ,  $T$ )
8:   else if  $a = x$  or  $a = y$  or  $a = z$  then
9:      $T \leftarrow$  VARTREE( $a$ ); PUSH( $\mathcal{S}$ ,  $T$ )
10:  else if  $a = )$  then
11:     $T \leftarrow$  REDUCE( $\mathcal{S}$ ); PUSH( $\mathcal{S}$ ,  $T$ )
12:  end if
13:   $a \leftarrow$  next symbol of text
14: end while
```

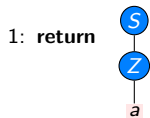

Ein Parser für die Grammatik \mathcal{G}_T

PARSE(*text*)

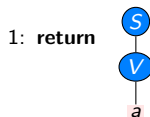
```
1:  $a \leftarrow$  first symbol of text
2: initialise stack  $\mathcal{S}$ 
3: while  $a \neq$  end of file do
4:   if  $a = ($  or  $a = +$  or  $a = *$  or  $a = -$  then
5:     PUSH( $\mathcal{S}, a$ )
6:   else if  $a = 0$  or  $a = 1$  then
7:      $T \leftarrow$  NUMTREE( $a$ ); PUSH( $\mathcal{S}, T$ )
8:   else if  $a = x$  or  $a = y$  or  $a = z$  then
9:      $T \leftarrow$  VARTREE( $a$ ); PUSH( $\mathcal{S}, T$ )
10:  else if  $a = )$  then
11:     $T \leftarrow$  REDUCE( $\mathcal{S}$ ); PUSH( $\mathcal{S}, T$ )
12:  end if
13:   $a \leftarrow$  next symbol of text
14: end while
15:  $T \leftarrow$  POP( $\mathcal{S}$ )
16: if  $T$  is a tree and  $\mathcal{S} = \emptyset$  then
17:   return  $T$ 
18: end if
```

Ein Parser für die Grammatik G_T (Forts.)

NUMTREE(a)

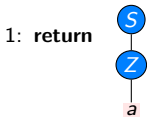


VARTREE(a)

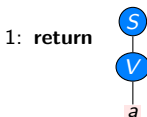


Ein Parser für die Grammatik G_T (Forts.)

NUMTREE(a)



VARTREE(a)

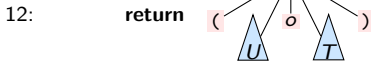


REDUCE(\mathcal{S})

- 1: $T \leftarrow \text{POP}(\mathcal{S})$
- 2: **if** T is a tree **then**
- 3: $o \leftarrow \text{POP}(\mathcal{S})$
- 4: **if** $o = -$ **then**
- 5: $p \leftarrow \text{POP}(\mathcal{S})$
- 6: **if** $p = ($ **then**



- 8: **end if**
- 9: **else if** $o = +$ or $o = *$ **then**
- 10: $U \leftarrow \text{POP}(\mathcal{S}); p \leftarrow \text{POP}(\mathcal{S})$
- 11: **if** U is a tree and $p = ($ **then**



- 13: **end if**
- 14: **end if**
- 15: **end if**

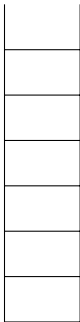
Beispiel 9.2

Eingabewort

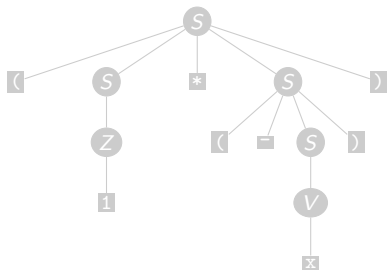
(1 * (- x))



Stack



Ableitungsbaum

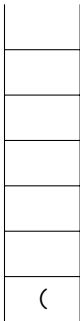


Eingabewort

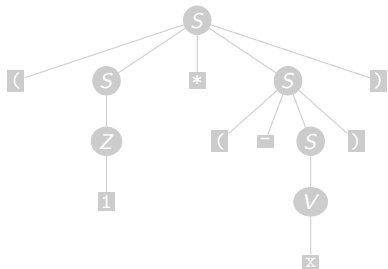
(1 * (- x))



Stack



Ableitungsbaum



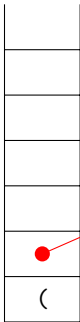
Beispiel 9.2

Eingabewort

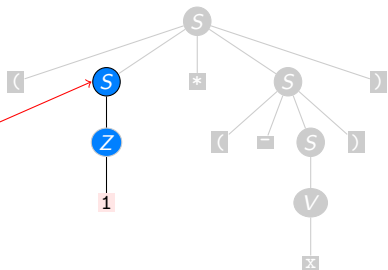
(1 * (- x))



Stack



Ableitungsbaum



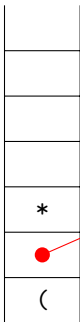
Beispiel 9.2

Eingabewort

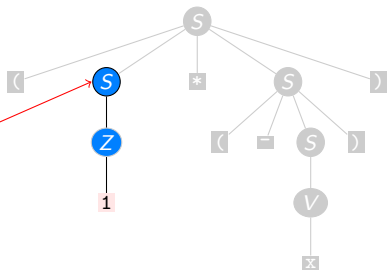
(1 * (- x))



Stack



Ableitungsbaum



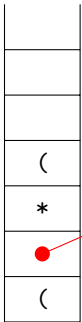
Beispiel 9.2

Eingabewort

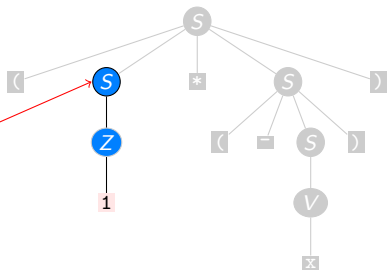
(1 * (- x))



Stack



Ableitungsbaum



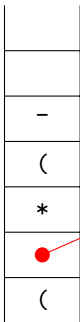
Beispiel 9.2

Eingabewort

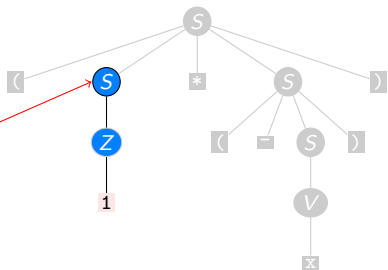
(1 * (- x))



Stack



Ableitungsbaum



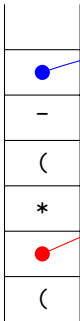
Beispiel 9.2

Eingabewort

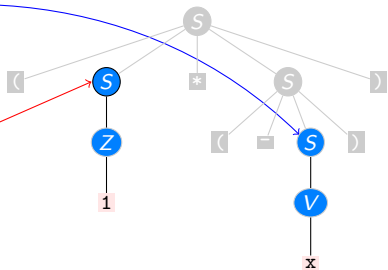
(1 * (- x))



Stack



Ableitungsbaum



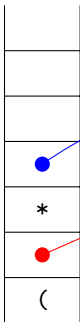
Beispiel 9.2

Eingabewort

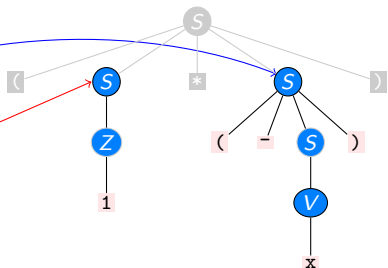
(1 * (- x))



Stack



Ableitungsbaum

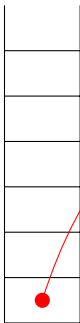


Eingabewort

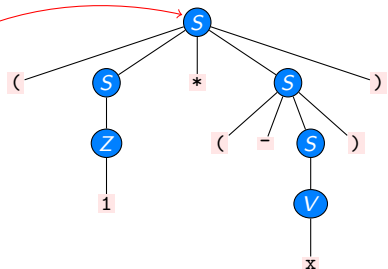
(1 * (- x))



Stack



Ableitungsbaum



Beobachtung 9.3

Unser Parser für die Grammatik \mathcal{G}_T arbeitet im Prinzip wie ein deterministischer Kellerautomat (DPDA) für die Sprache L_T .

DPDA für die arithmetischen Terme

Folgender DPDA erkennt die Sprache $L_{T\$} := \{x\$ \mid x \in L_T\}$.

$\mathcal{A} = (\{q_0, q_1, \dots, q_6\}, \Sigma \cup \{\$\}, \Gamma, \Delta, q_0, Z_0, \{q_6\})$, wobei

$$\Gamma = \{Z_0, T, X_(), X_+, X_*, X_-\}$$

und Δ besteht aus folgenden Transitionen:

Stackaufbau

$(q_0, (, *, q_0, X_(*))$

$(q_0, +, *, q_0, X_+(*))$

$(q_0, *, *, q_0, X_*(*))$

$(q_0, -, *, q_0, X_-(*))$ für $*$ $\in \Gamma$

Zahlen

$(q_0, 0, *, q_0, T_*)$

$(q_0, 1, *, q_0, T_*)$ für $*$ $\in \Gamma$

Variablen

$(q_0, x, *, q_0, T_*)$

$(q_0, y, *, q_0, T_*)$

$(q_0, z, *, q_0, T_*)$ für $*$ $\in \Gamma$

Stackabbau (Reduce)

$(q_0,), T, q_1, \varepsilon)$

Minus

$(q_1, \varepsilon, X_-, q_2, \varepsilon)$

$(q_2, \varepsilon, X_(), q_0, T)$

Plus und Mal

$(q_1, \varepsilon, X_\circ, q_3, \varepsilon)$ für $\circ \in \{+, *\}$

$(q_3, \varepsilon, T, q_4, \varepsilon)$

$(q_4, \varepsilon, X_(), q_0, T)$

Akzeptanz

$(q_0, \$, T, q_5, \varepsilon)$

$(q_5, \varepsilon, Z_0, q_6, \varepsilon)$

- ▶ Unsere Grammatik für die arithmetischen Terme ist “leicht zu parsen”. Insbesondere ist sie **eindeutig**, und die Sprache der Terme ist **DPDA-erkennbar**.
- ▶ Unser Parser ist ein **LR-Parser**: der Text wird von **links** nach rechts gelesen, und die Hauptarbeit (Reduce) wird am **rechten** Ende der Regeln gemacht.
- ▶ Manche Grammatiken sind noch einfacher und erlauben **LL-Parser**, bei denen man schon beim Lesen des ersten Symbols der Regel (also am **linken** Ende) entscheidet, wie zu verfahren ist.
- ▶ Die Theorie kontextfreier Sprachen und deterministischer PDAs ist so weit entwickelt, dass man Parser in der Regel automatisch generieren kann.

Mehr dazu in der Vorlesung **Compilerbau**.

Abschnitt 9.2

Das Wortproblem für kontextfreie Grammatiken

Der CYK-Algorithmus (Cocke-Younger-Kasami) erlaubt es, Ableitungen von Wörtern in kontextfreien Grammatiken in Chomsky-Normalform zu finden.

Es ist dabei nicht notwendig, dass die Grammatiken eindeutig sind.

Der Einfachheit halber betrachten wir nur das Entscheidungsproblem:

Gegeben eine kontextfreie Grammatik \mathcal{G} und ein Wort w , entscheide, ob $w \in L(\mathcal{G})$

(Das **Wortproblem** für kontextfreie Grammatiken.)

Eingabe

Kontextfreie Grammatik $\mathcal{G} = (N, \Sigma, P, S)$ in CNF, Wort $w = a_1 \dots a_n \in \Sigma^*$

Algorithmus

1. Berechne für alle i, j nach wachsender Distanz $d = j - i$ die Mengen

$$N_{ij} = \{A \in N \mid A \xrightarrow{*} \underbrace{w[i, j]}_{=a_i a_{i+1} \dots a_j}\}.$$

2. Bestimme, ob $S \in N_{1n}$.

Die Berechnung der N_{ij} erfolgt induktiv über d .

Induktive Berechnung der N_{ij}

Induktionsanfang $d = 0$ (also $i = j$)

$$A \in N_{ii} \iff A \xrightarrow{*} a_i \iff A \rightarrow a_i \in P.$$

Induktionsschritt $d > 0$ (also $i < j$)

$$A \in N_{ij} \iff A \xrightarrow{*} w[i, j]$$

\iff es gibt eine Regel $A \rightarrow BC$ und ein k mit $i \leq k < j$, so dass

$$B \xrightarrow{*} w[i, k] \quad \text{und} \quad C \xrightarrow{*} w[k+1, j]$$

\iff es gibt eine Regel $A \rightarrow BC$ und ein k mit $i \leq k < j$, so dass

$$B \in N_{ik} \quad \text{und} \quad C \in N_{k+1j}.$$

Grammatik \mathcal{G} :

$$S \rightarrow SA \mid a, \quad A \rightarrow BS, \quad B \rightarrow BB \mid BS \mid b \mid c$$

Eingabewort $w = abaabca$

Tabelle für die N_{ij} -Werte

Beispiel 9.4

Grammatik \mathcal{G} :

$$S \rightarrow SA \mid a, \quad A \rightarrow BS, \quad B \rightarrow BB \mid BS \mid b \mid c$$

Eingabewort $w = abaabca$

Tabelle für die N_{ij} -Werte

<i>a</i>	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}	N_{16}	N_{17}
	<i>b</i>	N_{22}	N_{23}	N_{24}	N_{25}	N_{26}	N_{27}
		<i>a</i>	N_{33}	N_{34}	N_{35}	N_{36}	N_{37}
			<i>a</i>	N_{44}	N_{45}	N_{46}	N_{47}
				<i>b</i>	N_{55}	N_{56}	N_{57}
					<i>c</i>	N_{66}	N_{67}
						<i>a</i>	N_{77}

Beispiel 9.4

Grammatik \mathcal{G} :

$$S \rightarrow SA \mid a, \quad A \rightarrow BS, \quad B \rightarrow BB \mid BS \mid b \mid c$$

Eingabewort $w = abaabca$

Tabelle für die N_{ij} -Werte

<i>a</i>	<i>S</i>	N_{12}	N_{13}	N_{14}	N_{15}	N_{16}	N_{17}
	<i>b</i>	<i>B</i>	N_{23}	N_{24}	N_{25}	N_{26}	N_{27}
		<i>a</i>	<i>S</i>	N_{34}	N_{35}	N_{36}	N_{37}
			<i>a</i>	<i>S</i>	N_{45}	N_{46}	N_{47}
				<i>b</i>	<i>B</i>	N_{56}	N_{57}
					<i>c</i>	<i>B</i>	N_{67}
						<i>a</i>	<i>S</i>

Beispiel 9.4

Grammatik \mathcal{G} :

$$S \rightarrow SA \mid a, \quad A \rightarrow BS, \quad B \rightarrow BB \mid BS \mid b \mid c$$

Eingabewort $w = abaabca$

Tabelle für die N_{ij} -Werte

a	S		N_{13}	N_{14}	N_{15}	N_{16}	N_{17}
	b	B	A, B	N_{24}	N_{25}	N_{26}	N_{27}
		a	S		N_{35}	N_{36}	N_{37}
			a	S		N_{46}	N_{47}
				b	B	B	N_{57}
					c	B	A, B
						a	S

Beispiel 9.4

Grammatik \mathcal{G} :

$$S \rightarrow SA \mid a, \quad A \rightarrow BS, \quad B \rightarrow BB \mid BS \mid b \mid c$$

Eingabewort $w = abaabca$

Tabelle für die N_{ij} -Werte

a	S		S	N_{14}	N_{15}	N_{16}	N_{17}
	b	B	A, B	A, B	N_{25}	N_{26}	N_{27}
		a	S			N_{36}	N_{37}
			a	S			N_{47}
				b	B	B	A, B
					c	B	A, B
						a	S

Beispiel 9.4

Grammatik \mathcal{G} :

$$S \rightarrow SA \mid a, \quad A \rightarrow BS, \quad B \rightarrow BB \mid BS \mid b \mid c$$

Eingabewort $w = abaabca$

Tabelle für die N_{ij} -Werte

a	S		S	S	N_{15}	N_{16}	N_{17}
	b	B	A, B	A, B	B	N_{26}	N_{27}
		a	S				N_{37}
			a	S			S
				b	B	B	A, B
					c	B	A, B
						a	S

Beispiel 9.4

Grammatik \mathcal{G} :

$$S \rightarrow SA \mid a, \quad A \rightarrow BS, \quad B \rightarrow BB \mid BS \mid b \mid c$$

Eingabewort $w = abaabca$

Tabelle für die N_{ij} -Werte

a	S		S	S		N_{16}	N_{17}
	b	B	A, B	A, B	B	B	N_{27}
		a	S				
			a	S			S
				b	B	B	A, B
					c	B	A, B
						a	S

Beispiel 9.4

Grammatik \mathcal{G} :

$$S \rightarrow SA \mid a, \quad A \rightarrow BS, \quad B \rightarrow BB \mid BS \mid b \mid c$$

Eingabewort $w = abaabca$

Tabelle für die N_{ij} -Werte

a	S		S	S			N_{17}
	b	B	A, B	A, B	B	B	A, B
		a	S				
			a	S			S
				b	B	B	A, B
					c	B	A, B
						a	S

Beispiel 9.4

Grammatik \mathcal{G} :

$$S \rightarrow SA \mid a, \quad A \rightarrow BS, \quad B \rightarrow BB \mid BS \mid b \mid c$$

Eingabewort $w = abaabca$

Tabelle für die N_{ij} -Werte

a	S		S	S			S
	b	B	A, B	A, B	B	B	A, B
		a	S				
			a	S			S
				b	B	B	A, B
					c	B	A, B
						a	S

Beispiel 9.4

Grammatik \mathcal{G} :

$$S \rightarrow SA \mid a, \quad A \rightarrow BS, \quad B \rightarrow BB \mid BS \mid b \mid c$$

Eingabewort $w = abaabca$

Tabelle für die N_{ij} -Werte

a	S		S	S			S
	b	B	A, B	A, B	B	B	A, B
		a	S				
			a	S			S
				b	B	B	A, B
					c	B	A, B
						a	S

Ableitung von w

$$\begin{aligned}
 & S \vdash SA \vdash SAA \vdash aAA \vdash aBSA \vdash aBSSA \vdash abSSA \vdash abaSA \\
 & \vdash abaaA \vdash abaaBS \vdash abaaBBS \vdash abaabBS \vdash abaabcS \vdash abaabca.
 \end{aligned}$$

Eingabe: $\mathcal{G} = (N, \Sigma, P, S)$ kontextfrei in CNF, $w = a_1 \dots a_n \in \Sigma^*$

- 1: **for** $i = 1$ to n **do**
- 2: $N_{ii} \leftarrow \{A \in N \mid A \rightarrow a_i \in P\}$
- 3: **end for**

Eingabe: $\mathcal{G} = (N, \Sigma, P, S)$ kontextfrei in CNF, $w = a_1 \dots a_n \in \Sigma^*$

- 1: **for** $i = 1$ to n **do**
- 2: $N_{ii} \leftarrow \{A \in N \mid A \rightarrow a_i \in P\}$
- 3: **end for**
- 4: **for** $d = 1$ to $n - 1$ **do**
- 5: **for** $i = 1$ to $n - d$ **do**
- 6: $j \leftarrow i + d$

Eingabe: $\mathcal{G} = (N, \Sigma, P, S)$ kontextfrei in CNF, $w = a_1 \dots a_n \in \Sigma^*$

```
1: for  $i = 1$  to  $n$  do
2:    $N_{ii} \leftarrow \{A \in N \mid A \rightarrow a_i \in P\}$ 
3: end for
4: for  $d = 1$  to  $n - 1$  do
5:   for  $i = 1$  to  $n - d$  do
6:      $j \leftarrow i + d$ 
7:      $N_{ij} \leftarrow \emptyset$ 
8:     for  $k = i$  to  $j - 1$  do
9:        $N_{ij} \leftarrow N_{ij} \cup \{A \mid A \rightarrow BC \in P \text{ with } B \in N_{ik} \text{ and } C \in N_{k+1j}\}$ 
10:    end for
11:  end for
12: end for
```


Eingabe: $\mathcal{G} = (N, \Sigma, P, S)$ kontextfrei in CNF, $w = a_1 \dots a_n \in \Sigma^*$

```
1: for  $i = 1$  to  $n$  do
2:    $N_{ii} \leftarrow \{A \in N \mid A \rightarrow a_i \in P\}$ 
3: end for
4: for  $d = 1$  to  $n - 1$  do
5:   for  $i = 1$  to  $n - d$  do
6:      $j \leftarrow i + d$ 
7:      $N_{ij} \leftarrow \emptyset$ 
8:     for  $k = i$  to  $j - 1$  do
9:        $N_{ij} \leftarrow N_{ij} \cup \{A \mid A \rightarrow BC \in P \text{ with } B \in N_{ik} \text{ and } C \in N_{k+1j}\}$ 
10:    end for
11:  end for
12: end for
13: if  $S \in N_{1n}$  then
14:   accept
15: else
16:   reject
17: end if
```

Zeitaufwand (grobe obere Abschätzung) bei festem \mathcal{G} .

- ▶ Initialisierung: $O(n)$
- ▶ Innere Schleife über k -Werte: $O(n)$
- ▶ Mittlere Schleife über i -Werte: $O(n \cdot n)$
- ▶ Äußere Schleife über d -Werte: $O(n \cdot n \cdot n)$

Insgesamt $O(n) + O(n^3)$, d.h. $O(n^3)$.

Satz 9.5

Der CYK-Algorithmus löst das Wortproblem für kontextfreie Grammatiken (in CNF) in kubischer Zeit in der Länge des betrachteten Wortes.

Abschnitt 9.3

Grenzen der algorithmischen Lösbarkeit

Grundlegende algorithmische Probleme

Wie bei den endlichen Automaten betrachten wir folgende grundlegende algorithmische Probleme, hier für kontextfreie Grammatiken:

1. **Wortproblem:** Gegeben kontextfreie Grammatik \mathcal{G} , Wort w .
Ist $w \in L(\mathcal{G})$?
2. **Leerheitsproblem:** Gegeben kontextfreie Grammatik \mathcal{G} .
Ist $L(\mathcal{G}) = \emptyset$?
3. **Unendlichkeitsproblem:** Gegeben kontextfreie Grammatik \mathcal{G} .
Ist $L(\mathcal{G})$ unendlich?
4. **Inklusionsproblem:** Gegeben kontextfreie Grammatiken $\mathcal{G}_1, \mathcal{G}_2$.
Gilt $L(\mathcal{G}_1) \subseteq L(\mathcal{G}_2)$?
5. **Äquivalenzproblem:** Gegeben kontextfreie Grammatiken $\mathcal{G}_1, \mathcal{G}_2$.
Gilt $L(\mathcal{G}_1) = L(\mathcal{G}_2)$?

Das Wortproblem für kontextfreie Grammatiken lässt sich mit Hilfe des CYK-Algorithmus folgendermaßen lösen.

Eingabe

Kontextfreie Grammatik \mathcal{G} , Wort $w \in \Sigma^* \setminus \{\varepsilon\}$.

Algorithmus

1. Konstruiere zu \mathcal{G} äquivalente Grammatik \mathcal{G}' in CNF.
2. Entscheide $w \in L(\mathcal{G}')$ mit Hilfe des CYK-Algorithmus.

Weil sich die Umwandlung einer Grammatik in CNF in Polynomialzeit durchführen lässt (vgl. Beweis von Satz 7.20), ist die Laufzeit insgesamt polynomiell.

Bemerkung 9.6

Mit einem ähnlichen Verfahren wie dem zur Elimination von ε -Regeln (vgl. Lemma 7.17) kann man auch in Polynomialzeit entscheiden, ob eine Grammatik das leere Wort erzeugt.

Beispiel 9.7

Grammatik \mathcal{G}

$S \rightarrow AaB \mid aB$

$A \rightarrow AA \mid Sbb$

$B \rightarrow Scc \mid A \mid DD$

$C \rightarrow EaS \mid SS$

$D \rightarrow SAB \mid bE$

$E \rightarrow aab$

Ist $L(\mathcal{G}) = \emptyset$?

Beispiel 9.7

Grammatik \mathcal{G}

$$S \rightarrow AaB \mid aB$$

$$A \rightarrow AA \mid Sbb$$

$$B \rightarrow Scc \mid A \mid DD$$

$$C \rightarrow EaS \mid SS$$

$$D \rightarrow SAB \mid bE$$

$$E \rightarrow aab$$

Ist $L(\mathcal{G}) = \emptyset$?

Nein!

$$S \rightarrow aB \rightarrow aDD \rightarrow abED \rightarrow abaabD \rightarrow abaabbE \rightarrow abaabbaab.$$

Definition 9.8

Sei $\mathcal{G} = (N, \Sigma, P, S)$ eine kontextfreie Grammatik.

Ein Nichtterminalsymbol $A \in N$ ist **terminierend** in \mathcal{G} , wenn es ein Wort $w \in \Sigma^*$ gibt, so dass $A \xrightarrow{*} w$.

Beobachtung 9.9

Sei $\mathcal{G} = (N, \Sigma, P, S)$ eine kontextfreie Grammatik.

Dann gilt

$$L(\mathcal{G}) \neq \emptyset \iff S \text{ ist terminierend.}$$

Eingabe

Kontextfreie Grammatik \mathcal{G} .

Ausgabe

Liste der terminierenden Nichtterminale von \mathcal{G} .

Algorithmus

1. Markiere auf den rechten Regelseiten alle Terminale.
2. Solange Regel mit unmarkierter linker Seite A existiert, so dass die rechte Seite voll markiert ist: markiere A überall.
3. Gib die Liste aller markierten Nichtterminale aus.

Grammatik \mathcal{G}

$S \rightarrow AaB \mid aB$

$A \rightarrow AA \mid Sbb$

$B \rightarrow Scc \mid A \mid DD$

$C \rightarrow EaS \mid SS$

$D \rightarrow SAB \mid bE$

$E \rightarrow aab$

Grammatik \mathcal{G}

$S \rightarrow A\underline{a}B \mid \underline{a}B$
 $A \rightarrow AA \mid S\underline{bb}$
 $B \rightarrow S\underline{cc} \mid A \mid DD$
 $C \rightarrow E\underline{a}S \mid SS$
 $D \rightarrow SAB \mid \underline{b}E$
 $E \rightarrow \underline{aab}$

Grammatik \mathcal{G}

$S \rightarrow A\underline{a}B \mid \underline{a}B$
 $A \rightarrow AA \mid S\underline{bb}$
 $B \rightarrow S\underline{cc} \mid A \mid DD$
 $C \rightarrow \underline{E}aS \mid SS$
 $D \rightarrow SAB \mid \underline{bE}$
 $\underline{E} \rightarrow \underline{aab}$

Grammatik \mathcal{G}

$S \rightarrow A\underline{a}B \mid \underline{a}B$

$A \rightarrow AA \mid S\underline{bb}$

$B \rightarrow S\underline{cc} \mid A \mid \underline{DD}$

$C \rightarrow \underline{E}aS \mid SS$

$\underline{D} \rightarrow SAB \mid \underline{bE}$

$\underline{E} \rightarrow \underline{aab}$

Grammatik \mathcal{G}

$S \rightarrow \underline{AaB} \mid \underline{aB}$
 $A \rightarrow AA \mid S\underline{bb}$
 $\underline{B} \rightarrow S\underline{cc} \mid A \mid \underline{DD}$
 $C \rightarrow \underline{EaS} \mid SS$
 $\underline{D} \rightarrow \underline{SAB} \mid \underline{bE}$
 $\underline{E} \rightarrow \underline{aab}$

Grammatik \mathcal{G}

$$\begin{aligned}\underline{S} &\rightarrow \underline{AaB} \mid \underline{aB} \\ \underline{A} &\rightarrow \underline{AA} \mid \underline{Sbb} \\ \underline{B} &\rightarrow \underline{Scc} \mid \underline{A} \mid \underline{DD} \\ \underline{C} &\rightarrow \underline{EaS} \mid \underline{SS} \\ \underline{D} &\rightarrow \underline{SAB} \mid \underline{bE} \\ \underline{E} &\rightarrow \underline{aab}\end{aligned}$$

Grammatik \mathcal{G}

$$\begin{aligned}\underline{S} &\rightarrow \underline{AaB} \mid \underline{aB} \\ \underline{A} &\rightarrow \underline{AA} \mid \underline{Sbb} \\ \underline{B} &\rightarrow \underline{Scc} \mid \underline{A} \mid \underline{DD} \\ \underline{C} &\rightarrow \underline{EaS} \mid \underline{SS} \\ \underline{D} &\rightarrow \underline{SAB} \mid \underline{bE} \\ \underline{E} &\rightarrow \underline{aab}\end{aligned}$$

Grammatik \mathcal{G}

$$\begin{aligned}\underline{S} &\rightarrow \underline{AaB} \mid \underline{aB} \\ \underline{A} &\rightarrow \underline{AA} \mid \underline{Sbb} \\ \underline{B} &\rightarrow \underline{Scc} \mid \underline{A} \mid \underline{DD} \\ \underline{C} &\rightarrow \underline{EaS} \mid \underline{SS} \\ \underline{D} &\rightarrow \underline{SAB} \mid \underline{bE} \\ \underline{E} &\rightarrow \underline{aab}\end{aligned}$$

Grammatik \mathcal{G}

$$\begin{aligned}
 \underline{S} &\rightarrow \underline{AaB} \mid \underline{aB} \\
 \underline{A} &\rightarrow \underline{AA} \mid \underline{Sbb} \\
 \underline{B} &\rightarrow \underline{Scc} \mid \underline{A} \mid \underline{DD} \\
 \underline{C} &\rightarrow \underline{EaS} \mid \underline{SS} \\
 \underline{D} &\rightarrow \underline{SAB} \mid \underline{bE} \\
 \underline{E} &\rightarrow \underline{aab}
 \end{aligned}$$

Ergebnis: Alle Nichtterminale sind terminierend.
 Insbesondere ist S terminierend und damit $L(\mathcal{G}) \neq \emptyset$.

Korrektheit des Markierungsalgorithmus

Satz 9.10

Der Markierungsalgorithmus berechnet korrekt die terminierenden Nichtterminalsymbole der Eingabegrammatik.

Beweis (Skizze).

Sei $\mathcal{G} = (N, \Sigma, P, S)$ die Eingabegrammatik.

Der Algorithmus hält nach spätestens $|N|$ Durchläufen der Schleife in Schritt 2 an, weil in jedem Durchlauf ein neues Nichtterminal markiert wird.

Behauptung 1

Falls ein Nichtterminal $A \in N$ vom Algorithmus markiert wird, so ist A terminierend.

Diese Behauptung kann man per Induktion über die Anzahl der Durchläufe der Schleife in Schritt 2 des Algorithmus (also die Anzahl der bereits markierten Nichtterminale) beweisen.

Behauptung 2

Falls ein Nichtterminal $A \in N$ terminierend ist, wird es vom Algorithmus markiert.

Diese Aussage lässt sich per Induktion über die Länge einer Ableitung eines Terminalwortes aus A beweisen. □

Eine naive Implementierung des Markierungsalgorithmus läuft in quadratischer Zeit:

- ▶ Die Schleife in Schritt 2 muss höchstens $|N|$ mal durchlaufen werden.
- ▶ Jeder Durchlauf der Schleife benötigt Zeit $O(|P|)$.

Eine geschicktere Variante des Algorithmus läuft in Linearzeit.

Korollar 9.11

Das Leerheitsproblem für kontextfreie Grammatiken lässt sich in Linearzeit lösen.

Für das Unendlichkeitsproblem gibt es einen einfachen Algorithmus, der auf dem Pumping-Lemma beruht:

Pumping-Lemma für kontextfreie Sprachen:

Sei $L \subseteq \Sigma^*$ kontextfrei. Dann gibt es eine Zahl $n \geq 1$, so dass jedes Wort $z \in L$ mit $|z| \geq n$ zerlegbar ist in Wörter $u, v, w, x, y \in \Sigma^*$ mit $z = uvwxy$, die folgende Eigenschaften haben:

- (i) $vx \neq \varepsilon$,
- (ii) $|vwx| \leq n$,
- (iii) für alle $k \geq 0$ gilt $uv^kwx^ky \in L$.

Anmerkung: Die Zahl n kann dabei als $n := 2^{|N|+1}$ gewählt werden, wobei N die Menge der Nichtterminale in einer CNF Grammatik für L ist. Mit etwas mehr Aufwand lässt sich das Problem auch in Polynomialzeit lösen.

- ▶ Das Äquivalenzproblem ist ein **unentscheidbares Problem**. Das heißt, dass es für dieses Problem überhaupt keinen Algorithmus gibt, der es (für alle Eingaben) löst.

Mehr dazu in der Vorlesung **Berechenbarkeit und Komplexität**.

- ▶ Das Inklusionsproblem ist ebenfalls **unentscheidbar**: Mit Hilfe des Inklusionsproblems könnte man ja das Äquivalenzproblem lösen: $L(G_1) = L(G_2)$ genau dann wenn $L(G_1) \subseteq L(G_2)$ und $L(G_2) \subseteq L(G_1)$.

Weitere unentscheidbare Probleme für kontextfreie Grammatiken

1. Universalitätsproblem

Gegeben Grammatik \mathcal{G} (mit Terminalalphabet Σ), entscheide, ob $L(\mathcal{G}) = \Sigma^*$.

2. Durchschnittsproblem

Gegeben zwei Grammatiken $\mathcal{G}, \mathcal{G}'$, entscheide, ob $L(\mathcal{G}) \cap L(\mathcal{G}') = \emptyset$.

3. Regularitätsproblem

Gegeben Grammatik \mathcal{G} , entscheide, ob $L(\mathcal{G})$ eine reguläre Sprache ist.

4. Eindeutigkeitsproblem

Gegeben Grammatik \mathcal{G} , entscheide, ob \mathcal{G} eindeutig ist, das heißt, ob jedes Wort in $L(\mathcal{G})$ einen eindeutigen Ableitungsbaum hat.