



Einführung in die Theoretische Informatik

David Drexel

Alexander Maringele

Julian Fodor

David Obwaller

Alexander Lochmann

Jonas Schöpf

Georg Moser

cbr.uibk.ac.at



Zusammenfassung

Zusammenfassung der letzten LVA

Definition (Deterministischer endlicher Automat (kurz: DEA))

Ein **DEA** ist ein 5-Tupel $A = (Q, \Sigma, \delta, q_0, F)$ sodass

- 1 Q eine endliche Menge von **Zuständen**
- 2 Σ eine endliche Menge von **Eingabesymbole**
- 3 $\delta: Q \times \Sigma \rightarrow Q$ die **Übergangsfunktion**
- 4 $q_0 \in Q$ der **Startzustand**
- 5 $F \subseteq Q$ eine endliche Menge von **akzeptierenden Zuständen**

Zu beachten: δ muss für alle möglichen Argumente definiert sein

Satz

Sei A ein DEA, dann ist $L(A)$ regulär und umgekehrt existiert zu jeder regulären Sprache L ein DEA A , sodass $L = L(A)$

Satz (für KFG)

Gelte $A \Rightarrow X_1 X_2 \dots X_n \xRightarrow{*} x$, wobei $X_i \in V \cup \Sigma$, $x \in \Sigma^*$; dann können wir x in die Stücke x_1, x_2, \dots, x_n brechen, sodass für alle i : $X_i \xRightarrow{*} x_i$

Beweis.

- 1 Wenn $X_i \in \Sigma$, dann $X_i = x_i$ und offensichtlich $X_i \xRightarrow{*} x_i$
- 2 Wenn $X_i \in V$, dann erhalten wir $X_i \xRightarrow{*} x_i$ indem
 - Expansionen von X_1, \dots, X_{i-1} eliminiert
 - Expansionen von X_{i+1}, \dots, X_n eliminiert
 - überflüssige Schritte weggelassen werden

Satz (für KFG)

Gelte $A \Rightarrow X_1 X_2 \dots X_n \xRightarrow{*} x$, wobei $X_i \in V \cup \Sigma$, $x \in \Sigma^*$; dann können wir x in die Stücke x_1, x_2, \dots, x_n brechen, sodass für alle i : $X_i \xRightarrow{*} x_i$

Beweis.

- 1 Wenn $X_i \in \Sigma$, dann $X_i = x_i$ und offensichtlich $X_i \xRightarrow{*} x_i$
- 2 Wenn $X_i \in V$, dann erhalten wir $X_i \xRightarrow{*} x_i$ indem
 - Expansionen von X_1, \dots, X_{i-1} eliminiert
 - Expansionen von X_{i+1}, \dots, X_n eliminiert
 - überflüssige Schritte weggelassen werden
- 3 Beachte, dass Expansionen von X_i immer **links von** Expansionen von X_j sind, wenn $i < j$; das gilt aber **nur** für KFGs

Einführung in die Logik

Syntax & Semantik der Aussagenlogik, Formales Beweisen, Konjunktive und Disjunktive Normalformen

Einführung in die Algebra

algebraische Strukturen, Beispiele von Algebren, Zusammenhang Boolesche Algebra und Aussagenlogik, Universelle Algebra, Satz von Birkhoff

Einführung in die Theorie der Formalen Sprachen

Grammatiken und Formale Sprachen, Chomsky-Hierarchie, Reguläre Sprachen, Kontextfreie Sprachen, Anwendungen von formalen Sprachen

Einführung in die Berechenbarkeitstheorie

Algorithmisch unlösbare Probleme, Turing Maschinen, Registermaschinen

Einführung in die Programmverifikation

Prinzipien der Analyse von Programmen, Verifikation nach Hoare

Einführung in die Logik

Syntax & Semantik der Aussagenlogik, Formales Beweisen, Konjunktive und Disjunktive Normalformen

Einführung in die Algebra

algebraische Strukturen, Beispiele von Algebren, Zusammenhang Boolesche Algebra und Aussagenlogik, Universelle Algebra, Satz von Birkhoff

Einführung in die Theorie der Formalen Sprachen

Grammatiken und Formale Sprachen, Chomsky-Hierarchie, Reguläre Sprachen, **Kontextfreie Sprachen, Anwendungen von formalen Sprachen**

Einführung in die Berechenbarkeitstheorie

Algorithmisch unlösbare Probleme, Turing Maschinen, Registermaschinen

Einführung in die Programmverifikation

Prinzipien der Analyse von Programmen, Verifikation nach Hoare

Definition

- Eine **Linksableitung** ist eine Ableitung sodass immer die am weitesten links stehende Variable ersetzt wird

 $\Rightarrow_l, \overset{*}{\Rightarrow}_l$

Definition

- Eine **Linksableitung** ist eine Ableitung sodass immer die am weitesten links stehende Variable ersetzt wird
- In einer **Rechtsableitung** wird immer die am weitesten rechts stehende Variable ersetzt

 $\Rightarrow_l, \Rightarrow_l^*$ $\Rightarrow_r, \Rightarrow_r^*$

Definition

- Eine **Linksableitung** ist eine Ableitung sodass immer die am weitesten links stehende Variable ersetzt wird
- In einer **Rechtsableitung** wird immer die am weitesten rechts stehende Variable ersetzt

 $\Rightarrow_l, \Rightarrow_l^*$ $\Rightarrow_r, \Rightarrow_r^*$

Beispiel

Wir betrachten KFG $G_2 = (\{S\}, \{(,)\}, R, S)$, wobei R :

$$S \rightarrow \epsilon \mid (S) \mid SS$$

Definition

- Eine **Linksableitung** ist eine Ableitung sodass immer die am weitesten links stehende Variable ersetzt wird
- In einer **Rechtsableitung** wird immer die am weitesten rechts stehende Variable ersetzt

 $\Rightarrow_l, \Rightarrow_l^*$ $\Rightarrow_r, \Rightarrow_r^*$

Beispiel

Wir betrachten KFG $G_2 = (\{S\}, \{(,)\}, R, S)$, wobei R :

$$S \rightarrow \epsilon \mid (S) \mid SS$$

dann gilt $S \Rightarrow_r SS \Rightarrow_r S(S) \Rightarrow_r S() \Rightarrow_r (S)() \Rightarrow_r ()()$

Definition

- Eine **Linksableitung** ist eine Ableitung sodass immer die am weitesten links stehende Variable ersetzt wird
- In einer **Rechtsableitung** wird immer die am weitesten rechts stehende Variable ersetzt

 $\Rightarrow_l, \overset{*}{\Rightarrow}_l$ $\Rightarrow_r, \overset{*}{\Rightarrow}_r$

Beispiel

Wir betrachten KFG $G_2 = (\{S\}, \{(,)\}, R, S)$, wobei R :

$$S \rightarrow \epsilon \mid (S) \mid SS$$

dann gilt $S \underset{r}{\Rightarrow} SS \underset{r}{\Rightarrow} S(S) \underset{r}{\Rightarrow} S() \underset{r}{\Rightarrow} (S)() \underset{r}{\Rightarrow} ()()$

Definition (Eindeutigkeit einer Grammatik)

KFG G heißt **eindeutig**, wenn jedes Wort $x \in L(G)$ genau eine Linksableitung besitzt, ansonsten **mehrdeutig**

Definition

Sei $A \rightarrow x \in R$ mit $x \in (V \cup \Sigma)^*$

Definition

Sei $A \rightarrow x \in R$ mit $x \in (V \cup \Sigma)^*$

Die Bestimmung der **Sprache $L(A)$ von A** mittels **rekursiver Inferenz**:

Definition

Sei $A \rightarrow x \in R$ mit $x \in (V \cup \Sigma)^*$

Die Bestimmung der **Sprache $L(A)$ von A** mittels **rekursiver Inferenz**:

- 1 Wenn $x \in \Sigma^*$, dann $x \in L(A)$

Definition

Sei $A \rightarrow x \in R$ mit $x \in (V \cup \Sigma)^*$

Die Bestimmung der **Sprache $L(A)$** von A mittels **rekursiver Inferenz**:

- 1 Wenn $x \in \Sigma^*$, dann **$x \in L(A)$**
- 2 Andererseits, sei $x = X_1 \cdots X_n$, wobei $X_i \in V \cup \Sigma$

Gelte außerdem $x_i \in L(X_i)$ oder $x_i = X_i \in \Sigma$

Dann gilt **$x_1 x_2 \cdots x_n \in L(A)$**

Definition

Sei $A \rightarrow x \in R$ mit $x \in (V \cup \Sigma)^*$

Die Bestimmung der **Sprache $L(A)$ von A** mittels **rekursiver Inferenz**:

- 1 Wenn $x \in \Sigma^*$, dann $x \in L(A)$
- 2 Andererseits, sei $x = X_1 \cdots X_n$, wobei $X_i \in V \cup \Sigma$

Gelte außerdem $x_i \in L(X_i)$ oder $x_i = X_i \in \Sigma$

Dann gilt $x_1 x_2 \cdots x_n \in L(A)$

Bemerkung

- Die rekursive Inferenz entspricht dem **bottom-up parsing** eines Compilers: zuerst wird der Rumpf einer Regel gelesen, dann wird das Ergebnis der Variable im Kopf übergeben
- Der Parsergenerator yacc generiert einen bottom-up parser (nach der Pause)

Beispiel

Wir betrachten die KFG $G_3 = (\{E, I\}, \{+, \cdot, (,)\}, a, b, 0, 1\}, R, E)$, wobei

$$\begin{array}{llll} E \rightarrow I & E \rightarrow E \cdot E & I \rightarrow a & I \rightarrow I0 \\ E \rightarrow E+E & E \rightarrow (E) & I \rightarrow Ib & I \rightarrow I1 \\ & I \rightarrow Ia & I \rightarrow b & \end{array}$$

Beispiel

Wir betrachten die KFG $G_3 = (\{E, I\}, \{+, \cdot, (,), a, b, 0, 1\}, R, E)$, wobei

$$\begin{array}{llll} E \rightarrow I & E \rightarrow E \cdot E & I \rightarrow a & I \rightarrow I0 \\ E \rightarrow E+E & E \rightarrow (E) & I \rightarrow Ib & I \rightarrow I1 \\ & I \rightarrow Ia & I \rightarrow b & \end{array}$$

und zeigen $(a+b10) \in L(E)$:

Beispiel

Wir betrachten die KFG $G_3 = (\{E, I\}, \{+, \cdot, (,), a, b, 0, 1\}, R, E)$, wobei

$$\begin{array}{llll} E \rightarrow I & E \rightarrow E \cdot E & I \rightarrow a & I \rightarrow I0 \\ E \rightarrow E+E & E \rightarrow (E) & I \rightarrow Ib & I \rightarrow I1 \\ & I \rightarrow Ia & I \rightarrow b & \end{array}$$

und zeigen $(a+b10) \in L(E)$:

	Wort x	Variable	Regel	Rekursion
1	a	I	$I \rightarrow a$	

Beispiel

Wir betrachten die KFG $G_3 = (\{E, I\}, \{+, \cdot, (,), a, b, 0, 1\}, R, E)$, wobei

$$\begin{array}{llll} E \rightarrow I & E \rightarrow E \cdot E & I \rightarrow a & I \rightarrow I0 \\ E \rightarrow E+E & E \rightarrow (E) & I \rightarrow Ib & I \rightarrow I1 \\ & I \rightarrow Ia & I \rightarrow b & \end{array}$$

und zeigen $(a+b10) \in L(E)$:

	Wort x	Variable	Regel	Rekursion
1	a	I	$I \rightarrow a$	
2	b	I	$I \rightarrow b$	

Beispiel

Wir betrachten die KFG $G_3 = (\{E, I\}, \{+, \cdot, (,), a, b, 0, 1\}, R, E)$, wobei

$$\begin{array}{llll} E \rightarrow I & E \rightarrow E \cdot E & I \rightarrow a & I \rightarrow I0 \\ E \rightarrow E+E & E \rightarrow (E) & I \rightarrow Ib & I \rightarrow I1 \\ & I \rightarrow Ia & I \rightarrow b & \end{array}$$

und zeigen $(a+b10) \in L(E)$:

	Wort x	Variable	Regel	Rekursion
1	a	I	$I \rightarrow a$	
2	b	I	$I \rightarrow b$	
3	b1	I	$I \rightarrow I1$	2

Beispiel

Wir betrachten die KFG $G_3 = (\{E, I\}, \{+, \cdot, (,), a, b, 0, 1\}, R, E)$, wobei

$$\begin{array}{llll} E \rightarrow I & E \rightarrow E \cdot E & I \rightarrow a & I \rightarrow I0 \\ E \rightarrow E+E & E \rightarrow (E) & I \rightarrow Ib & I \rightarrow I1 \\ & I \rightarrow Ia & I \rightarrow b & \end{array}$$

und zeigen $(a+b10) \in L(E)$:

	Wort x	Variable	Regel	Rekursion
1	a	I	$I \rightarrow a$	
2	b	I	$I \rightarrow b$	
3	b1	I	$I \rightarrow I1$	2
4	b10	I	$I \rightarrow I0$	3

Beispiel

Wir betrachten die KFG $G_3 = (\{E, I\}, \{+, \cdot, (,), a, b, 0, 1\}, R, E)$, wobei

$$\begin{array}{llll} E \rightarrow I & E \rightarrow E \cdot E & I \rightarrow a & I \rightarrow I0 \\ E \rightarrow E+E & E \rightarrow (E) & I \rightarrow Ib & I \rightarrow I1 \\ & I \rightarrow Ia & I \rightarrow b & \end{array}$$

und zeigen $(a+b10) \in L(E)$:

	Wort x	Variable	Regel	Rekursion
1	a	I	$I \rightarrow a$	
2	b	I	$I \rightarrow b$	
3	b1	I	$I \rightarrow I1$	2
4	b10	I	$I \rightarrow I0$	3
5	a	E	$E \rightarrow I$	1
6	b10	E	$E \rightarrow I$	4

Beispiel

Wir betrachten die KFG $G_3 = (\{E, I\}, \{+, \cdot, (,), a, b, 0, 1\}, R, E)$, wobei

$$\begin{array}{llll} E \rightarrow I & E \rightarrow E \cdot E & I \rightarrow a & I \rightarrow I0 \\ E \rightarrow E+E & E \rightarrow (E) & I \rightarrow Ib & I \rightarrow I1 \\ & I \rightarrow Ia & I \rightarrow b & \end{array}$$

und zeigen $(a+b10) \in L(E)$:

	Wort x	Variable	Regel	Rekursion
1	a	I	$I \rightarrow a$	
2	b	I	$I \rightarrow b$	
3	b1	I	$I \rightarrow I1$	2
4	b10	I	$I \rightarrow I0$	3
5	a	E	$E \rightarrow I$	1
6	b10	E	$E \rightarrow I$	4
7	a+b10	E	$E \rightarrow E+E$	5,6

Beispiel

Wir betrachten die KFG $G_3 = (\{E, I\}, \{+, \cdot, (,), a, b, 0, 1\}, R, E)$, wobei

$$\begin{array}{llll} E \rightarrow I & E \rightarrow E \cdot E & I \rightarrow a & I \rightarrow I0 \\ E \rightarrow E+E & E \rightarrow (E) & I \rightarrow Ib & I \rightarrow I1 \\ & I \rightarrow Ia & I \rightarrow b & \end{array}$$

und zeigen $(a+b10) \in L(E)$:

	Wort x	Variable	Regel	Rekursion
1	a	I	$I \rightarrow a$	
2	b	I	$I \rightarrow b$	
3	b1	I	$I \rightarrow I1$	2
4	b10	I	$I \rightarrow I0$	3
5	a	E	$E \rightarrow I$	1
6	b10	E	$E \rightarrow I$	4
7	a+b10	E	$E \rightarrow E+E$	5, 6
8	$(a+b10)$	E	$E \rightarrow (E)$	7

Beispiel

Wir betrachten die KFG $G_3 = (\{E, I\}, \{+, \cdot, (,), a, b, 0, 1\}, R, E)$, wobei

$$\begin{array}{llll} E \rightarrow I & E \rightarrow E \cdot E & I \rightarrow a & I \rightarrow I0 \\ E \rightarrow E+E & E \rightarrow (E) & I \rightarrow Ib & I \rightarrow I1 \\ & I \rightarrow Ia & I \rightarrow b & \end{array}$$

und zeigen $(a+b10) \in L(E)$:

	Wort x	Variable	Regel	Rekursion
1	a	I	$I \rightarrow a$	
2	b	I	$I \rightarrow b$	
3	b1	I	$I \rightarrow I1$	2
4	b10	I	$I \rightarrow I0$	3
5	a	E	$E \rightarrow I$	1
6	b10	E	$E \rightarrow I$	4
7	a+b10	E	$E \rightarrow E+E$	5, 6
8	$(a+b10)$	E	$E \rightarrow (E)$	7

Definition (Syntaxbaum)

Ein **Syntaxbaum** für KFG $G = (V, \Sigma, R, S)$ ist ein Baum B mit:

Definition (Syntaxbaum)

Ein **Syntaxbaum** für KFG $G = (V, \Sigma, R, S)$ ist ein Baum B mit:

- 1 Jeder innere Knoten von B ist eine Variable in V

Definition (Syntaxbaum)

Ein **Syntaxbaum** für KFG $G = (V, \Sigma, R, S)$ ist ein Baum B mit:

- 1 Jeder innere Knoten von B ist eine Variable in V
- 2 Jedes Blatt in B ist entweder:
 - ein Terminal aus Σ
 - ein Nichtterminal aus V , oder
 - ϵ

Im letzten Fall ist das Blatt das einzige Kind seines Vorgängers

Definition (Syntaxbaum)

Ein **Syntaxbaum** für KFG $G = (V, \Sigma, R, S)$ ist ein Baum B mit:

- 1 Jeder innere Knoten von B ist eine Variable in V
- 2 Jedes Blatt in B ist entweder:
 - ein Terminal aus Σ
 - ein Nichtterminal aus V , oder
 - ϵ

Im letzten Fall ist das Blatt das einzige Kind seines Vorgängers

- 3 Sei A ein innerer Knoten, X_1, \dots, X_n seine Kinder ($X_i \in V \cup \Sigma$); dann gilt:

$$A \rightarrow X_1 \cdots X_n \in R$$

Definition (Syntaxbaum)

Ein **Syntaxbaum** für KFG $G = (V, \Sigma, R, S)$ ist ein Baum B mit:

- 1 Jeder innere Knoten von B ist eine Variable in V
- 2 Jedes Blatt in B ist entweder:
 - ein Terminal aus Σ
 - ein Nichtterminal aus V , oder
 - ϵ

Im letzten Fall ist das Blatt das einzige Kind seines Vorgängers

- 3 Sei A ein innerer Knoten, X_1, \dots, X_n seine Kinder ($X_i \in V \cup \Sigma$); dann gilt:

$$A \rightarrow X_1 \cdots X_n \in R$$

Das **Ergebnis** eines Syntaxbaums B für G ist das Wort über $(V \cup \Sigma)^*$, das wir erhalten, wenn wir die Blätter in B von links nach rechts lesen

Beispiel

Wir betrachten die KFG $G = (\{P\}, \Sigma, R, P)$ mit:

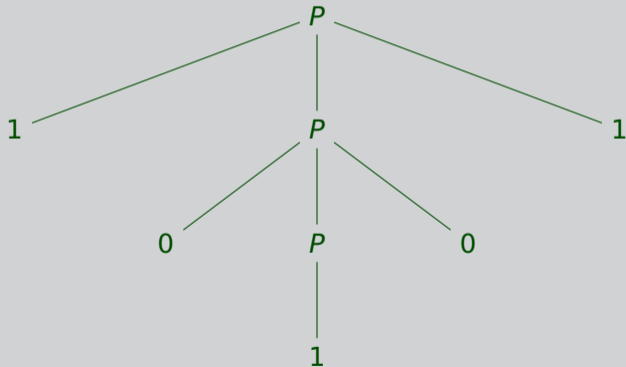
$$P \rightarrow \epsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$$

Beispiel

Wir betrachten die KFG $G = (\{P\}, \Sigma, R, P)$ mit:

$$P \rightarrow \epsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$$

Dann ist der folgende Baum ein Syntaxbaum für G_1 :

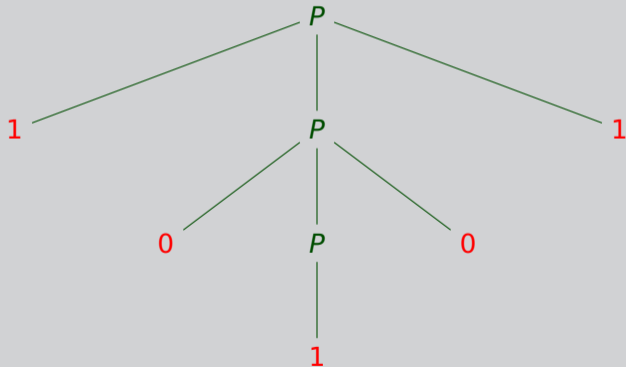


Beispiel

Wir betrachten die KFG $G = (\{P\}, \Sigma, R, P)$ mit:

$$P \rightarrow \epsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$$

Dann ist der folgende Baum ein Syntaxbaum für G_1 :



Satz

Sei Σ ein Alphabet und sei $x \in \Sigma^*$. Die folgenden Beschreibungen kontextfreier Sprachen sind **äquivalent**:

1 $x \in L(A)$ nach dem rekursiven Inferenzverfahren

2 $A \xRightarrow{*} x$

3 $A \xRightarrow[\ell]{*} x$

4 $A \xRightarrow[r]{*} x$

5 Es existiert ein Syntaxbaum mit Wurzel A und Ergebnis x

Satz

Sei Σ ein Alphabet und sei $x \in \Sigma^*$. Die folgenden Beschreibungen kontextfreier Sprachen sind **äquivalent**:

1 $x \in L(A)$ nach dem rekursiven Inferenzverfahren

2 $A \xRightarrow{*} x$

3 $A \xRightarrow[\ell]{*} x$

4 $A \xRightarrow[r]{*} x$

5 Es existiert ein Syntaxbaum mit Wurzel A und Ergebnis x

$x \in L(A)$

\exists Syntaxbaum

$A \xRightarrow{*} x$

$A \xRightarrow[\ell]{*} x$

Satz

Sei Σ ein Alphabet und sei $x \in \Sigma^*$. Die folgenden Beschreibungen kontextfreier Sprachen sind **äquivalent**:

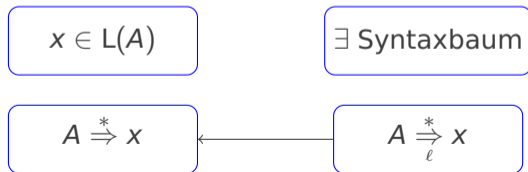
1 $x \in L(A)$ nach dem rekursiven Inferenzverfahren

2 $A \xRightarrow{*} x$

3 $A \xRightarrow[\ell]{*} x$

4 $A \xRightarrow[r]{*} x$

5 Es existiert ein Syntaxbaum mit Wurzel A und Ergebnis x



Satz

Sei Σ ein Alphabet und sei $x \in \Sigma^*$. Die folgenden Beschreibungen kontextfreier Sprachen sind **äquivalent**:

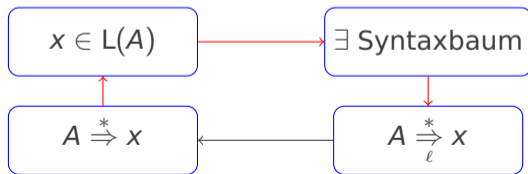
1 $x \in L(A)$ nach dem rekursiven Inferenzverfahren

2 $A \xRightarrow{*} x$

3 $A \xRightarrow[\ell]{*} x$

4 $A \xRightarrow[r]{*} x$

5 Es existiert ein Syntaxbaum mit Wurzel A und Ergebnis x



Satz

Angenommen $x \in L(A)$ nach dem rekursiven Inferenzverfahren, dann gibt es einen Syntaxbaum mit Wurzel A und Ergebnis x

Satz

Angenommen $x \in L(A)$ nach dem rekursiven Inferenzverfahren, dann gibt es einen Syntaxbaum mit Wurzel A und Ergebnis x

Beweis.

Mit Induktion nach der Anzahl der Rekursionen im Inferenzverfahren

Satz

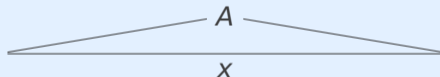
Angenommen $x \in L(A)$ nach dem rekursiven Inferenzverfahren, dann gibt es einen Syntaxbaum mit Wurzel A und Ergebnis x

Beweis.

Mit Induktion nach der Anzahl der Rekursionen im Inferenzverfahren

1 **Basis** $x \in L(A)$ benutzt genau die Regel $A \rightarrow x \in R$

Dann \exists Syntaxbaum mit Wurzel A :



Satz

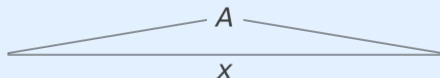
Angenommen $x \in L(A)$ nach dem rekursiven Inferenzverfahren, dann gibt es einen Syntaxbaum mit Wurzel A und Ergebnis x

Beweis.

Mit Induktion nach der Anzahl der Rekursionen im Inferenzverfahren

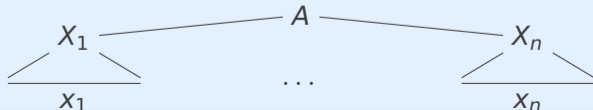
- 1 **Basis** $x \in L(A)$ benutzt genau die Regel $A \rightarrow x \in R$

Dann \exists Syntaxbaum mit Wurzel A :



- 2 **Schritt** $x \in L(A)$ benutzt $A \rightarrow X_1 \cdots X_n$ ($X_i \in V \cup \Sigma$)

Dann \exists Syntaxbaum mit Wurzel A :



Satz

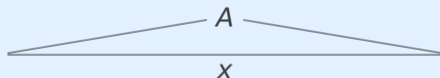
Angenommen $x \in L(A)$ nach dem rekursiven Inferenzverfahren, dann gibt es einen Syntaxbaum mit Wurzel A und Ergebnis x

Beweis.

Mit Induktion nach der Anzahl der Rekursionen im Inferenzverfahren

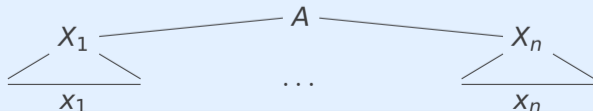
- 1 **Basis** $x \in L(A)$ benutzt genau die Regel $A \rightarrow x \in R$

Dann \exists Syntaxbaum mit Wurzel A :



- 2 **Schritt** $x \in L(A)$ benutzt $A \rightarrow X_1 \cdots X_n$ ($X_i \in V \cup \Sigma$)

Dann \exists Syntaxbaum mit Wurzel A :



Satz

Sei S ein Syntaxbaum mit Wurzel A und Ergebnis $x \in \Sigma^$, dann gibt es eine Linksableitung (eine Rechtsableitung) von x aus A*

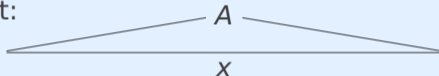
Satz

Sei S ein Syntaxbaum mit Wurzel A und Ergebnis $x \in \Sigma^*$, dann gibt es eine Linksableitung (eine Rechtsableitung) von x aus A

Beweis.

Mit Induktion nach der Höhe des Syntaxbaums

1 Basis Habe S die Gestalt:



dann existiert $A \rightarrow x \in R$, also $A \xRightarrow{\ell} x$

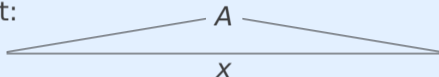
Satz

Sei S ein Syntaxbaum mit Wurzel A und Ergebnis $x \in \Sigma^*$, dann gibt es eine Linksableitung (eine Rechtsableitung) von x aus A

Beweis.

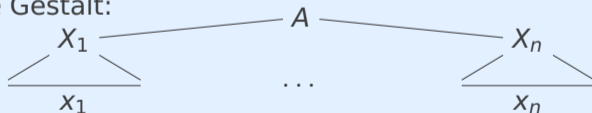
Mit Induktion nach der Höhe des Syntaxbaums

1 Basis Habe S die Gestalt:



dann existiert $A \rightarrow x \in R$, also $A \xRightarrow{\ell} x$

2 Schritt Habe S die Gestalt:



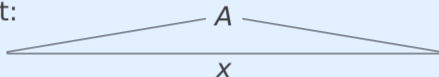
Satz

Sei S ein Syntaxbaum mit Wurzel A und Ergebnis $x \in \Sigma^*$, dann gibt es eine Linksableitung (eine Rechtsableitung) von x aus A

Beweis.

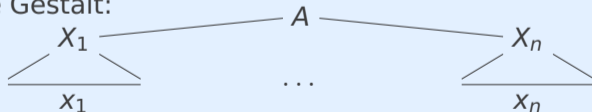
Mit Induktion nach der Höhe des Syntaxbaums

1 Basis Habe S die Gestalt:



dann existiert $A \rightarrow x \in R$, also $A \xRightarrow{\ell} x$

2 Schritt Habe S die Gestalt:



dann $\exists A \rightarrow X_1 \cdots X_n \in R$, also:

$$A \xRightarrow{\ell} X_1 \cdots X_n \xRightarrow{\ell}^* x_1 x_2 \cdots x_n \xRightarrow{\ell}^* x_1 \cdots x_{n-1} x_n$$

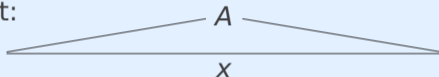
Satz

Sei S ein Syntaxbaum mit Wurzel A und Ergebnis $x \in \Sigma^*$, dann gibt es eine Linksableitung (eine Rechtsableitung) von x aus A

Beweis.

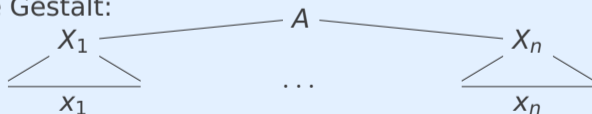
Mit Induktion nach der Höhe des Syntaxbaums

1 **Basis** Habe S die Gestalt:



dann existiert $A \rightarrow x \in R$, also $A \xRightarrow{\ell} x$

2 **Schritt** Habe S die Gestalt:



dann $\exists A \rightarrow X_1 \cdots X_n \in R$, also:

$$A \xRightarrow{\ell} X_1 \cdots X_n \xRightarrow{\ell}^* x_1 x_2 \cdots x_n \xRightarrow{\ell}^* x_1 \cdots x_{n-1} x_n$$

Satz

Angenommen $A \stackrel{*}{\Rightarrow} x$ mit $x \in \Sigma^*$, dann liefert das rekursive Inferenzverfahren, dass $x \in L(A)$

Satz

Angenommen $A \xRightarrow{*} x$ mit $x \in \Sigma^*$, dann liefert das rekursive Inferenzverfahren, dass $x \in L(A)$

Beweis.

Mit Induktion nach der Länge ℓ der Ableitung $A \xRightarrow{*} x$:

Satz

Angenommen $A \xRightarrow{*} x$ mit $x \in \Sigma^*$, dann liefert das rekursive Inferenzverfahren, dass $x \in L(A)$

Beweis.

Mit Induktion nach der Länge ℓ der Ableitung $A \xRightarrow{*} x$:

1 Basis Sei $\ell = 1$, dann gilt $x \in L(A)$

Satz

Angenommen $A \Rightarrow^* x$ mit $x \in \Sigma^*$, dann liefert das rekursive Inferenzverfahren, dass $x \in L(A)$

Beweis.

Mit Induktion nach der Länge ℓ der Ableitung $A \Rightarrow^* x$:

- 1 Basis** Sei $\ell = 1$, dann gilt $x \in L(A)$
- 2 Schritt** Angenommen $\ell = \ell' + 1$

Satz

Angenommen $A \Rightarrow^* x$ mit $x \in \Sigma^*$, dann liefert das rekursive Inferenzverfahren, dass $x \in L(A)$

Beweis.

Mit Induktion nach der Länge ℓ der Ableitung $A \Rightarrow^* x$:

1 Basis Sei $\ell = 1$, dann gilt $x \in L(A)$

2 Schritt Angenommen $\ell = \ell' + 1$

Zunächst können wir die Ableitung $A \Rightarrow^* x$ wie folgt schreiben:

$$A \Rightarrow X_1 X_2 \cdots X_n \Rightarrow^* x$$

Satz

Angenommen $A \Rightarrow^* x$ mit $x \in \Sigma^*$, dann liefert das rekursive Inferenzverfahren, dass $x \in L(A)$

Beweis.

Mit Induktion nach der Länge ℓ der Ableitung $A \Rightarrow^* x$:

1 Basis Sei $\ell = 1$, dann gilt $x \in L(A)$

2 Schritt Angenommen $\ell = \ell' + 1$

Zunächst können wir die Ableitung $A \Rightarrow^* x$ wie folgt schreiben:

$$A \Rightarrow X_1 X_2 \cdots X_n \Rightarrow^* x = x_1 x_2 \cdots x_n$$

Satz

Angenommen $A \Rightarrow^* x$ mit $x \in \Sigma^*$, dann liefert das rekursive Inferenzverfahren, dass $x \in L(A)$

Beweis.

Mit Induktion nach der Länge ℓ der Ableitung $A \Rightarrow^* x$:

1 Basis Sei $\ell = 1$, dann gilt $x \in L(A)$

2 Schritt Angenommen $\ell = \ell' + 1$

Zunächst können wir die Ableitung $A \Rightarrow^* x$ wie folgt schreiben:

$$A \Rightarrow X_1 X_2 \cdots X_n \Rightarrow^* x = x_1 x_2 \cdots x_n$$

Wir verwenden, dass wir die Ableitungen der Sätze x_i aufbrechen können,

Satz

Angenommen $A \xRightarrow{*} x$ mit $x \in \Sigma^*$, dann liefert das rekursive Inferenzverfahren, dass $x \in L(A)$

Beweis.

Mit Induktion nach der Länge ℓ der Ableitung $A \xRightarrow{*} x$:

1 Basis Sei $\ell = 1$, dann gilt $x \in L(A)$

2 Schritt Angenommen $\ell = \ell' + 1$

Zunächst können wir die Ableitung $A \xRightarrow{*} x$ wie folgt schreiben:

$$A \Rightarrow X_1 X_2 \cdots X_n \xRightarrow{*} x = x_1 x_2 \cdots x_n$$

Wir verwenden, dass wir die Ableitungen der Sätze x_i aufbrechen können, also gilt:

- Wenn $X_i \in \Sigma$, dann $X_i = x_i$
- Wenn $X_i \in V$, dann gilt $X_i \xRightarrow{*} x_i$ und somit $x_i \in L(X_i)$

Satz

Angenommen $A \xRightarrow{*} x$ mit $x \in \Sigma^*$, dann liefert das rekursive Inferenzverfahren, dass $x \in L(A)$

Beweis.

Mit Induktion nach der Länge ℓ der Ableitung $A \xRightarrow{*} x$:

1 Basis Sei $\ell = 1$, dann gilt $x \in L(A)$

2 Schritt Angenommen $\ell = \ell' + 1$

Zunächst können wir die Ableitung $A \xRightarrow{*} x$ wie folgt schreiben:

$$A \Rightarrow X_1 X_2 \cdots X_n \xRightarrow{*} x = x_1 x_2 \cdots x_n$$

Wir verwenden, dass wir die Ableitungen der Sätze x_i aufbrechen können, also gilt:

- Wenn $X_i \in \Sigma$, dann $X_i = x_i$
- Wenn $X_i \in V$, dann gilt $X_i \xRightarrow{*} x_i$ und somit $x_i \in L(X_i)$

Anwendung der Theorie der Kontextfreien Sprachen

- Parsergeneratoren, beziehungsweise im **Compilerbau**
 - 1 Parsergenerator verwandelt die Beschreibung einer Sprache in einen Parser für diese Sprache
 - 2 Parser werden zur syntaktischen Analyse von Programmen verwendet

Anwendung der Theorie der Kontextfreien Sprachen

- Parsergeneratoren, beziehungsweise im **Compilerbau**
 - 1 Parsergenerator verwandelt die Beschreibung einer Sprache in einen Parser für diese Sprache
 - 2 Parser werden zur syntaktischen Analyse von Programmen verwendet
- Anwendungen im Bereich der **Wissensrepräsentation**, etwa in XML-Dokumenten (siehe Skriptum)

lexikalische Analyse

- der Eingabestrom aus ASCII-Zeichen wird analysiert
- in terminale Symbole der formalen Sprache zerlegt, die man **Token** nennt

Beispiel

Betrachte die Eingabe eines (simplen) Taschenrechners, dann sind die Token die Ziffern der Rechnung, charakterisiert etwa durch den folgenden **regulären Ausdruck**

$$([0-9]^+ | ([0-9]^* \backslash . [0-9]^+)) ([eE] [-+]? [0-9]^+)?$$

mögliche Eingabe: $1.0 + (2.1e1 - 3 * 5) * 0.5$

Beispiel

Betrachte die Eingabe eines (simplen) Taschenrechners, dann sind die Token die Ziffern der Rechnung, charakterisiert etwa durch den folgenden **regulären Ausdruck**

$$([0-9]+ | ([0-9]* \. [0-9]+)) ([eE] [-+]? [0-9]+)?$$

mögliche Eingabe: $1.0 + (2.1e1 - 3 * 5) * 0.5$

die Regeln der lexikalischen Analyse werden in der folgenden Form definiert

Muster **Aktion**

Lexikalische Analyse in C

Regelteil calc.l

```
%%  
([0-9]+|([0-9]*\.[0-9]+))([eE] [-+]?[0-9]+)? {  
    yy1val.dval = atof(yytext);  
    return NUMBER;  
}  
%%
```

Lexikalische Analyse in C

Regelteil calc.l

```
%%  
([0-9]+|([0-9]*\.[0-9]+))([eE][+-]?[0-9]+)? {  
    yylval.dval = atof(yytext);  
    return NUMBER;  
}  
%%
```

Parsergenerierung mit `lex` & `yacc`

Generell bestehen die Hilfsdateien des Parsergenerators aus

- einem **Definitionsteil**,
- einem **Regelteil**, der zwischen `%%` eingeschlossen wird, und
- einem **Programmteil** für benutzereigene Funktionen.

Beispiel

Grammatik G für arithmetische Ausdrücke

$$E \rightarrow T \mid +T \mid -T \mid E+T \mid E-T$$

$$T \rightarrow F \mid T * F \mid T / F$$

$$F \rightarrow c \mid v \mid (E) .$$

Beispiel

Grammatik G für arithmetische Ausdrücke

$$E \rightarrow T \mid +T \mid -T \mid E+T \mid E-T$$

$$T \rightarrow F \mid T * F \mid T / F$$

$$F \rightarrow c \mid v \mid (E) .$$

in G gilt etwa $-c * v \in L(E)$:

Beispiel

Grammatik G für arithmetische Ausdrücke

$$E \rightarrow T \mid +T \mid -T \mid E+T \mid E-T$$

$$T \rightarrow F \mid T * F \mid T / F$$

$$F \rightarrow c \mid v \mid (E) .$$

in G gilt etwa $-c * v \in L(E)$:

	Wort x	Variable	Regel	Rekursion
1	c	F	$F \rightarrow c$	
2	v	F	$F \rightarrow v$	
3	c	T	$T \rightarrow F$	1
4	$c * v$	T	$T \rightarrow T * F$	3, 2
5	$-c * v$	E	$E \rightarrow -T$	4

Parsergenerierung in C mit yacc

Regelteil calc.y (gekürzt)

```
expression:      term          { $$ = $1; }
                | '+' term     { $$ = $2; }
                | '-' term     { $$ = -$2; }
                | expression '+' term { $$ = $1 + $3; }
                | expression '-' term { $$ = $1 - $3; }
                ;

term:            factor        { $$ = $1; }
                | term '*' factor { $$ = $1 * $3; }
                | term '/' factor { $$ = $1 / $3; }
                ;

factor:         NUMBER        { $$ = $1; }
                | '(' expression ')' { $$ = $2; }
                ;
```