

Vorlesung

Grundlagen der Theoretischen Informatik / Einführung in die Theoretische Informatik I

Bernhard Beckert

Institut für Informatik



Sommersemester 2007

Dank

Diese Vorlesungsmaterialien basieren ganz wesentlich auf den Folien zu den Vorlesungen von

Katrin Erk (gehalten an der Universität Koblenz-Landau)

Jürgen Dix (gehalten an der TU Clausthal)

Ihnen beiden gilt mein herzlicher Dank.

– *Bernhard Beckert, April 2007*

Teil II

Terminologie

- 1 Sprache, Grammatik
- 2 Warum Sprachen?
- 3 Die Chomsky-Hierarchie
- 4 Probleme über Sprachen
- 5 Endlich, unendlich und dann?

Inhalt von Teil II

In den folgenden Abschnitten führen wir die Begriffe

- **Sprache**
- **Grammatik**

ein.

Wir untersuchen insbesondere

- ① wie man Probleme aus der Mathematik, Graphentheorie, Logik als **Probleme über Sprachen** formulieren kann.
- ② wie man Klassen von Grammatiken von steigendem Schwierigkeitsgrad definiert: **Chomsky-Hierarchie**.
- ③ **wieviele** Grammatiken und Sprachen **es überhaupt gibt** (soviele wie natürliche Zahlen, reelle Zahlen oder komplexe Zahlen?)

Inhalt von Teil II

In den folgenden Abschnitten führen wir die Begriffe

- **Sprache**
- **Grammatik**

ein.

Wir untersuchen insbesondere

- 1 **wie man Probleme aus der Mathematik, Graphentheorie, Logik als Probleme über Sprachen formulieren kann.**
- 2 wie man Klassen von Grammatiken von steigendem Schwierigkeitsgrad definiert: **Chomsky-Hierarchie.**
- 3 **wieviele** Grammatiken und Sprachen **es überhaupt gibt** (soviele wie natürliche Zahlen, reelle Zahlen oder komplexe Zahlen?)

Inhalt von Teil II

In den folgenden Abschnitten führen wir die Begriffe

- **Sprache**
- **Grammatik**

ein.

Wir untersuchen insbesondere

- 1 wie man Probleme aus der Mathematik, Graphentheorie, Logik als **Probleme über Sprachen** formulieren kann.
- 2 wie man Klassen von Grammatiken von steigendem Schwierigkeitsgrad definiert: **Chomsky-Hierarchie**.
- 3 **wieviele** Grammatiken und Sprachen **es überhaupt gibt**
(soviele wie natürliche Zahlen, reelle Zahlen oder komplexe Zahlen?)

Inhalt von Teil II

In den folgenden Abschnitten führen wir die Begriffe

- **Sprache**
- **Grammatik**

ein.

Wir untersuchen insbesondere

- 1 wie man Probleme aus der Mathematik, Graphentheorie, Logik als **Probleme über Sprachen** formulieren kann.
- 2 wie man Klassen von Grammatiken von steigendem Schwierigkeitsgrad definiert: **Chomsky-Hierarchie**.
- 3 **wieviele Grammatiken und Sprachen es überhaupt gibt**
(soviele wie natürliche Zahlen, reelle Zahlen oder komplexe Zahlen?)

Teil II

Terminologie

- 1 Sprache, Grammatik
- 2 Warum Sprachen?
- 3 Die Chomsky-Hierarchie
- 4 Probleme über Sprachen
- 5 Endlich, unendlich und dann?

Teil II

Terminologie

- 1 Sprache, Grammatik**
- 2 Warum Sprachen?
- 3 Die Chomsky-Hierarchie
- 4 Probleme über Sprachen
- 5 Endlich, unendlich und dann?

Definition 6.1 (Alphabet)

Ein **Alphabet** ist eine Menge von Zeichen/Buchstaben

- Grundlage einer Sprache (die zur Verfügung stehenden Zeichen)
- Meist endlich

Definition 6.2 (Wort)

Ein **Wort** (über einem Alphabet Σ)
ist eine endliche Folge von Zeichen aus Σ

$|w|$ bezeichnet Länge eines Wortes w

ε bezeichnet das leere Wort

Definition 6.1 (Alphabet)

Ein **Alphabet** ist eine Menge von Zeichen/Buchstaben

- Grundlage einer Sprache (die zur Verfügung stehenden Zeichen)
- Meist endlich

Definition 6.2 (Wort)

Ein **Wort** (über einem Alphabet Σ)
ist eine endliche Folge von Zeichen aus Σ

$|w|$ bezeichnet Länge eines Wortes w

ε bezeichnet das leere Wort

Definition 6.1 (Alphabet)

Ein **Alphabet** ist eine Menge von Zeichen/Buchstaben

- Grundlage einer Sprache (die zur Verfügung stehenden Zeichen)
- **Meist endlich**

Definition 6.2 (Wort)

Ein **Wort** (über einem Alphabet Σ)
ist eine endliche Folge von Zeichen aus Σ

$|w|$ bezeichnet Länge eines Wortes w

ε bezeichnet das leere Wort

Definition 6.1 (Alphabet)

Ein **Alphabet** ist eine Menge von Zeichen/Buchstaben

- Grundlage einer Sprache (die zur Verfügung stehenden Zeichen)
- Meist endlich

Definition 6.2 (Wort)

Ein **Wort** (über einem Alphabet Σ)
ist eine endliche Folge von Zeichen aus Σ

$|w|$ bezeichnet Länge eines Wortes w

ε bezeichnet das leere Wort

Definition 6.1 (Alphabet)

Ein **Alphabet** ist eine Menge von Zeichen/Buchstaben

- Grundlage einer Sprache (die zur Verfügung stehenden Zeichen)
- Meist endlich

Definition 6.2 (Wort)

Ein **Wort** (über einem Alphabet Σ)
ist eine endliche Folge von Zeichen aus Σ

$|w|$ bezeichnet Länge eines Wortes w

ϵ bezeichnet das leere Wort

Definition 6.1 (Alphabet)

Ein **Alphabet** ist eine Menge von Zeichen/Buchstaben

- Grundlage einer Sprache (die zur Verfügung stehenden Zeichen)
- Meist endlich

Definition 6.2 (Wort)

Ein **Wort** (über einem Alphabet Σ)
ist eine endliche Folge von Zeichen aus Σ

$|w|$ bezeichnet Länge eines Wortes w

ε bezeichnet das leere Wort

Definition 6.1 (Alphabet)

Ein **Alphabet** ist eine Menge von Zeichen/Buchstaben

- Grundlage einer Sprache (die zur Verfügung stehenden Zeichen)
- Meist endlich

Definition 6.2 (Wort)

Ein **Wort** (über einem Alphabet Σ)
ist eine endliche Folge von Zeichen aus Σ

$|w|$ bezeichnet Länge eines Wortes w

ε bezeichnet das leere Wort

Operationen auf Wörtern

Verknüpfung (Konkatenation):

$$w \circ w'$$

assoziativ, oft geschrieben als ww'

i -te Potenz:

$$w^0 = \varepsilon, \quad w^{i+1} = ww^i$$

Reverse:

w^R = das Wort w rückwärts

Operationen auf Wörtern

Verknüpfung (Konkatenation):

$$w \circ w'$$

assoziativ, oft geschrieben als ww'

***i*-te Potenz:**

$$w^0 = \varepsilon, \quad w^{i+1} = ww^i$$

Reverse:

w^R = das Wort w rückwärts

Operationen auf Wörtern

Verknüpfung (Konkatenation):

$$w \circ w'$$

assoziativ, oft geschrieben als ww'

i-te Potenz:

$$w^0 = \varepsilon, \quad w^{i+1} = ww^i$$

Reverse:

w^R = das Wort w rückwärts

Definition 6.3 (Sprache)

Eine Sprache L (über einem Alphabet Σ)
ist eine Menge von Wörtern über Σ .

Operationen auf Sprachen

Konkatenation:

$$L \circ M = \{w \circ w' \mid w \in L, w' \in M\}$$

n -te Potenz:

$$L^0 = \{\varepsilon\}, \quad L^{i+1} := LL^i$$

Reverse:

$$L^R = \{w^R \mid w \in L\}$$

Definition 6.3 (Sprache)

Eine Sprache L (über einem Alphabet Σ) ist eine Menge von Wörtern über Σ .

Operationen auf Sprachen

Konkatenation:

$$L \circ M = \{w \circ w' \mid w \in L, w' \in M\}$$

i -te Potenz:

$$L^0 = \{\varepsilon\}, \quad L^{i+1} := LL^i$$

Reverse:

$$L^R = \{w^R \mid w \in L\}$$

Definition 6.3 (Sprache)

Eine Sprache L (über einem Alphabet Σ) ist eine Menge von Wörtern über Σ .

Operationen auf Sprachen

Konkatenation:

$$L \circ M = \{w \circ w' \mid w \in L, w' \in M\}$$

i-te Potenz:

$$L^0 = \{\varepsilon\}, \quad L^{i+1} := LL^i$$

Reverse:

$$L^R = \{w^R : w \in L\}$$

Definition 6.3 (Sprache)

Eine Sprache L (über einem Alphabet Σ) ist eine Menge von Wörtern über Σ .

Operationen auf Sprachen

Konkatenation:

$$L \circ M = \{w \circ w' \mid w \in L, w' \in M\}$$

i -te Potenz:

$$L^0 = \{\varepsilon\}, \quad L^{i+1} := LL^i$$

Reverse:

$$L^R = \{w^R : w \in L\}$$

Kleene-Hülle

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$$

Variante:

$$L^+ = LL^* = L^1 \cup L^2 \cup \dots$$

Σ^* bezeichnet die Menge aller Wörter über Σ

Genau genommen besteht ein Unterschied:

ein Buchstabe \neq Wort, das nur aus dem einen Buchstaben besteht

Darum ist Σ selbst keine Sprache über Σ

(Oft wird über diesen Unterschied hinweggesehen)

Kleene-Hülle

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$$

Variante:

$$L^+ = LL^* = L^1 \cup L^2 \cup \dots$$

Σ^* bezeichnet die Menge aller Wörter über Σ

Genau genommen besteht ein Unterschied:

ein Buchstabe \neq Wort, das nur aus dem einen Buchstaben besteht

Darum ist Σ selbst keine Sprache über Σ

(Oft wird über diesen Unterschied hinweggesehen)

Kleene-Hülle

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$$

Variante:

$$L^+ = LL^* = L^1 \cup L^2 \cup \dots$$

Σ^* bezeichnet die Menge aller Wörter über Σ

Genau genommen besteht ein Unterschied:

ein Buchstabe \neq Wort, das nur aus dem einen Buchstaben besteht

Darum ist Σ selbst keine Sprache über Σ

(Oft wird über diesen Unterschied hinweggesehen)

Kleene-Hülle

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$$

Variante:

$$L^+ = LL^* = L^1 \cup L^2 \cup \dots$$

Σ^* bezeichnet die Menge aller Wörter über Σ

Genau genommen besteht ein Unterschied:

ein Buchstabe \neq Wort, das nur aus dem einen Buchstaben besteht

Darum ist Σ selbst keine Sprache über Σ

(Oft wird über diesen Unterschied hinweggesehen)

Kleene-Hülle

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$$

Variante:

$$L^+ = LL^* = L^1 \cup L^2 \cup \dots$$

Σ^* bezeichnet die Menge aller Wörter über Σ

Genau genommen besteht ein Unterschied:

ein Buchstabe \neq Wort, das nur aus dem einen Buchstaben besteht

Darum ist Σ selbst keine Sprache über Σ

(Oft wird über diesen Unterschied hinweggesehen)

Kleene-Hülle

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$$

Variante:

$$L^+ = LL^* = L^1 \cup L^2 \cup \dots$$

Σ^* bezeichnet die Menge aller Wörter über Σ

Genau genommen besteht ein Unterschied:

ein Buchstabe \neq Wort, das nur aus dem einen Buchstaben besteht

Darum ist Σ selbst keine Sprache über Σ

(Oft wird über diesen Unterschied hinweggesehen)

Definition 6.4 (Reguläre Ausdrücke)

Menge $\mathfrak{R}_{\text{reg}}_{\Sigma}$ der **regulären Ausdrücke** (über Σ) ist definiert durch:

- \emptyset ist ein regulärer Ausdruck
- Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck
- Sind r und s reguläre Ausdrücke, so auch

Klammern können weggelassen werden, dann

- $*$ hat Vorrang vor Konkatenation
- Konkatenation hat Vorrang vor $+$

Definition 6.4 (Reguläre Ausdrücke)

Menge \mathfrak{Reg}_Σ der **regulären Ausdrücke** (über Σ) ist definiert durch:

- 1 0 ist ein regulärer Ausdruck
- 2 Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck
- 3 Sind r und s reguläre Ausdrücke, so auch
 - $(r + s)$ (Vereinigung),
 - (rs) (Konkatenation),
 - (r^*) (Kleene Stern)

Klammern können weggelassen werden, dann

- * hat Vorrang vor Konkatenation
- Konkatenation hat Vorrang vor +

Definition 6.4 (Reguläre Ausdrücke)

Menge \mathfrak{Reg}_Σ der **regulären Ausdrücke** (über Σ) ist definiert durch:

- 1 0 ist ein regulärer Ausdruck
- 2 Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck
- 3 Sind r und s reguläre Ausdrücke, so auch
 - $(r + s)$ (Vereinigung),
 - (rs) (Konkatenation),
 - (r^*) (Kleene Stern)

Klammern können weggelassen werden, dann

- * hat Vorrang vor Konkatenation
- Konkatenation hat Vorrang vor +

Definition 6.4 (Reguläre Ausdrücke)

Menge \mathfrak{Reg}_Σ der **regulären Ausdrücke** (über Σ) ist definiert durch:

- 1 0 ist ein regulärer Ausdruck
- 2 Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck
- 3 Sind r und s reguläre Ausdrücke, so auch
 - $(r + s)$ (Vereinigung),
 - (rs) (Konkatenation),
 - (r^*) (Kleene Stern)

Klammern können weggelassen werden, dann

- * hat Vorrang vor Konkatenation
- Konkatenation hat Vorrang vor +

Definition 6.4 (Reguläre Ausdrücke)

Menge \mathfrak{Reg}_Σ der **regulären Ausdrücke** (über Σ) ist definiert durch:

- 1 0 ist ein regulärer Ausdruck
- 2 Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck
- 3 Sind r und s reguläre Ausdrücke, so auch
 - $(r + s)$ (Vereinigung),
 - (rs) (Konkatenation),
 - (r^*) (Kleene Stern)

Klammern können weggelassen werden, dann

- * hat Vorrang vor Konkatenation
- Konkatenation hat Vorrang vor +

Definition 6.4 (Reguläre Ausdrücke)

Menge \mathfrak{Reg}_Σ der **regulären Ausdrücke** (über Σ) ist definiert durch:

- 1 0 ist ein regulärer Ausdruck
- 2 Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck
- 3 Sind r und s reguläre Ausdrücke, so auch
 - $(r + s)$ (Vereinigung),
 - (rs) (Konkatenation),
 - (r^*) (Kleene Stern)

Klammern können weggelassen werden, dann

- $*$ hat Vorrang vor Konkatenation
- Konkatenation hat Vorrang vor $+$

Reguläre Ausdrücke

Definition 6.5 (Semantik regulärer Ausdrücke)

Ein regulärer Ausdruck r stellt eine Sprache $\mathcal{J}(r)$ über Σ wie folgt dar:

$$\mathcal{J}(0) := \emptyset$$

$$\mathcal{J}(a) := \{a\} \quad \text{für } a \in \Sigma$$

$$\mathcal{J}(r+s) := \mathcal{J}(r) \cup \mathcal{J}(s)$$

$$\mathcal{J}(rs) := \mathcal{J}(r)\mathcal{J}(s)$$

$$\mathcal{J}(r^*) := \mathcal{J}(r)^*$$

Wir benutzen auch das Makro ...

$$1 := 0^*$$

Es gilt: $\mathcal{J}(1) = \{\epsilon\}$

Reguläre Ausdrücke

Definition 6.5 (Semantik regulärer Ausdrücke)

Ein regulärer Ausdruck r stellt eine Sprache $\mathcal{J}(r)$ über Σ wie folgt dar:

$$\mathcal{J}(0) := \emptyset$$

$$\mathcal{J}(a) := \{a\} \quad \text{für } a \in \Sigma$$

$$\mathcal{J}(r+s) := \mathcal{J}(r) \cup \mathcal{J}(s)$$

$$\mathcal{J}(rs) := \mathcal{J}(r)\mathcal{J}(s)$$

$$\mathcal{J}(r^*) := \mathcal{J}(r)^*$$

Wir benutzen auch das Makro ...

$$1 := 0^*$$

Es gilt: $\mathcal{J}(1) = \{\epsilon\}$

Reguläre Ausdrücke

Definition 6.5 (Semantik regulärer Ausdrücke)

Ein regulärer Ausdruck r stellt eine Sprache $\mathcal{J}(r)$ über Σ wie folgt dar:

$$\mathcal{J}(0) := \emptyset$$

$$\mathcal{J}(a) := \{a\} \quad \text{für } a \in \Sigma$$

$$\mathcal{J}(r+s) := \mathcal{J}(r) \cup \mathcal{J}(s)$$

$$\mathcal{J}(rs) := \mathcal{J}(r)\mathcal{J}(s)$$

$$\mathcal{J}(r^*) := \mathcal{J}(r)^*$$

Wir benutzen auch das Makro ...

$$1 := 0^*$$

Es gilt: $\mathcal{J}(1) = \{\epsilon\}$

Reguläre Ausdrücke

Definition 6.5 (Semantik regulärer Ausdrücke)

Ein regulärer Ausdruck r stellt eine Sprache $\mathcal{J}(r)$ über Σ wie folgt dar:

$$\mathcal{J}(0) := \emptyset$$

$$\mathcal{J}(a) := \{a\} \quad \text{für } a \in \Sigma$$

$$\mathcal{J}(r+s) := \mathcal{J}(r) \cup \mathcal{J}(s)$$

$$\mathcal{J}(rs) := \mathcal{J}(r)\mathcal{J}(s)$$

$$\mathcal{J}(r^*) := \mathcal{J}(r)^*$$

Wir benutzen auch das Makro ...

$$1 := 0^*$$

Es gilt: $\mathcal{J}(1) = \{e\}$

Reguläre Ausdrücke

Definition 6.5 (Semantik regulärer Ausdrücke)

Ein regulärer Ausdruck r stellt eine Sprache $\mathcal{J}(r)$ über Σ wie folgt dar:

$$\mathcal{J}(0) := \emptyset$$

$$\mathcal{J}(a) := \{a\} \quad \text{für } a \in \Sigma$$

$$\mathcal{J}(r+s) := \mathcal{J}(r) \cup \mathcal{J}(s)$$

$$\mathcal{J}(rs) := \mathcal{J}(r)\mathcal{J}(s)$$

$$\mathcal{J}(r^*) := \mathcal{J}(r)^*$$

Wir benutzen auch das Makro ...

$$1 := 0^*$$

Es gilt: $\mathcal{J}(1) = \{\epsilon\}$

Reguläre Ausdrücke

Definition 6.5 (Semantik regulärer Ausdrücke)

Ein regulärer Ausdruck r stellt eine Sprache $\mathcal{J}(r)$ über Σ wie folgt dar:

$$\mathcal{J}(0) := \emptyset$$

$$\mathcal{J}(a) := \{a\} \quad \text{für } a \in \Sigma$$

$$\mathcal{J}(r+s) := \mathcal{J}(r) \cup \mathcal{J}(s)$$

$$\mathcal{J}(rs) := \mathcal{J}(r)\mathcal{J}(s)$$

$$\mathcal{J}(r^*) := \mathcal{J}(r)^*$$

Wir benutzen auch das Makro ...

$$1 := 0^*$$

Es gilt: $\mathcal{J}(1) = \{e\}$

Reguläre Ausdrücke

Definition 6.5 (Semantik regulärer Ausdrücke)

Ein regulärer Ausdruck r stellt eine Sprache $\mathcal{J}(r)$ über Σ wie folgt dar:

$$\mathcal{J}(0) := \emptyset$$

$$\mathcal{J}(a) := \{a\} \quad \text{für } a \in \Sigma$$

$$\mathcal{J}(r+s) := \mathcal{J}(r) \cup \mathcal{J}(s)$$

$$\mathcal{J}(rs) := \mathcal{J}(r)\mathcal{J}(s)$$

$$\mathcal{J}(r^*) := \mathcal{J}(r)^*$$

Wir benutzen auch das Makro ...

$$1 := 0^*$$

Es gilt: $\mathcal{J}(1) = \{\varepsilon\}$

Übung

Welche Sprachen werden durch die folgenden regulären Ausdrücke dargestellt?

- aa
- $(a + b)^*$
- $aa^* + bb^*$

Übung

Welche Sprachen werden durch die folgenden regulären Ausdrücke dargestellt?

- aa
- $(a + b)^*$
- $aa^* + bb^*$

Übung

Welche Sprachen werden durch die folgenden regulären Ausdrücke dargestellt?

- aa
- $(a + b)^*$
- $aa^* + bb^*$

Das Kommando grep (bzw. egrep)

- Sucht Wörter (Strings) in Dateien
- Benutzt reguläre Ausdrücke als Suchmuster
- Sehr schnell
- Volle Funktionalität mit egrep (UNIX/LINUX)

Das Kommando grep (bzw. egrep)

- Sucht Wörter (Strings) in Dateien
- **Benutzt reguläre Ausdrücke als Suchmuster**
- Sehr schnell
- Volle Funktionalität mit egrep (UNIX/LINUX)

Das Kommando grep (bzw. egrep)

- Sucht Wörter (Strings) in Dateien
- Benutzt reguläre Ausdrücke als Suchmuster
- **Sehr schnell**
- Volle Funktionalität mit egrep (UNIX/LINUX)

Das Kommando grep (bzw. egrep)

- Sucht Wörter (Strings) in Dateien
- Benutzt reguläre Ausdrücke als Suchmuster
- Sehr schnell
- **Volle Funktionalität mit egrep (UNIX/LINUX)**

Reguläre Ausdrücke als Suchmuster für grep

Syntax bei grep

grep	Regulärer Ausdruck
------	--------------------

ww'	ww'
-------	-------

$w w'$	$w + w'$
----------	----------

w^*	w^*
-------	-------

w^+	w^+
-------	-------

Syntactic Sugar

grep	Regulärer Ausdruck
------	--------------------

$[abc]$	$a + b + c$
---------	-------------

$[a-d]$	$a + b + c + d$
---------	-----------------

$.$	beliebiges Zeichen aus Σ
-----	---------------------------------

Reguläre Ausdrücke als Suchmuster für grep

Syntax bei grep

grep	Regulärer Ausdruck
------	--------------------

ww'	ww'
-------	-------

$w w'$	$w + w'$
----------	----------

w^*	w^*
-------	-------

w^+	w^+
-------	-------

Syntactic Sugar

grep	Regulärer Ausdruck
------	--------------------

$[abc]$	$a + b + c$
---------	-------------

$[a-d]$	$a + b + c + d$
---------	-----------------

$.$	beliebiges Zeichen aus Σ
-----	---------------------------------

Grammatik

- **Beschreibt eine Sprache**
- Menge von Regeln, mit deren Hilfe man Wörter ableiten kann
- Die zu einer Grammatik gehörende Sprache besteht aus den
 - **ableitbaren**
 - **terminalen**Wörtern

Grammatik

- Beschreibt eine Sprache
- Menge von Regeln, mit deren Hilfe man Wörter ableiten kann
- Die zu einer Grammatik gehörende Sprache besteht aus den
 - ableitbaren
 - terminalenWörtern

Grammatik

- Beschreibt eine Sprache
- Menge von Regeln, mit deren Hilfe man Wörter ableiten kann
- Die zu einer Grammatik gehörende Sprache besteht aus den
 - ableitbaren
 - terminalenWörtern

Definition 6.6 (Grammatik)

Eine **Grammatik** G über einem Alphabet Σ ist ein Tupel

$$G = (V, T, R, S)$$

Dabei ist

- V eine endliche Menge von **Variablen**
- $T \subseteq \Sigma$ eine endliche Menge von **Terminalen** mit $V \cap T = \emptyset$
- R eine **endliche** Menge von **Regeln**
- $S \in V$ das **Startsymbol**

Definition 6.6 (Grammatik)

Eine **Grammatik** G über einem Alphabet Σ ist ein Tupel

$$G = (V, T, R, S)$$

Dabei ist

- V eine endliche Menge von **Variablen**
- $T \subseteq \Sigma$ eine endliche Menge von **Terminalen** mit $V \cap T = \emptyset$
- R eine **endliche** Menge von **Regeln**
- $S \in V$ das **Startsymbol**

Definition 6.6 (Grammatik)

Eine **Grammatik** G über einem Alphabet Σ ist ein Tupel

$$G = (V, T, R, S)$$

Dabei ist

- V eine endliche Menge von **Variablen**
- $T \subseteq \Sigma$ eine endliche Menge von **Terminalen** mit $V \cap T = \emptyset$
- R eine **endliche** Menge von **Regeln**
- $S \in V$ das **Startsymbol**

Definition 6.6 (Grammatik)

Eine **Grammatik** G über einem Alphabet Σ ist ein Tupel

$$G = (V, T, R, S)$$

Dabei ist

- V eine endliche Menge von **Variablen**
- $T \subseteq \Sigma$ eine endliche Menge von **Terminalen** mit $V \cap T = \emptyset$
- R eine **endliche** Menge von **Regeln**
- $S \in V$ das **Startsymbol**

Definition 6.6 (Grammatik)

Eine **Grammatik** G über einem Alphabet Σ ist ein Tupel

$$G = (V, T, R, S)$$

Dabei ist

- V eine endliche Menge von **Variablen**
- $T \subseteq \Sigma$ eine endliche Menge von **Terminalen** mit $V \cap T = \emptyset$
- R eine **endliche** Menge von **Regeln**
- $S \in V$ das **Startsymbol**

Definition 6.7 (Regel)

Eine Regel ist ein Element

$$(P, Q) \in ((V \cup T)^* V (V \cup T)^*) \times (V \cup T)^*$$

Das heißt:

- P und Q sind Wörter über $(V \cup T)$
- P muss mindestens eine Variable enthalten
- Q ist beliebig

Bezeichnung:

P : Prämisse

Q : Conclusio

Definition 6.7 (Regel)

Eine Regel ist ein Element

$$(P, Q) \in ((V \cup T)^* V (V \cup T)^*) \times (V \cup T)^*$$

Das heißt:

- P und Q sind Wörter über $(V \cup T)$
- P muss mindestens eine Variable enthalten
- Q ist beliebig

Bezeichnung:

P : Prämisse

Q : Conclusio

Definition 6.7 (Regel)

Eine Regel ist ein Element

$$(P, Q) \in ((V \cup T)^* V (V \cup T)^*) \times (V \cup T)^*$$

Das heißt:

- P und Q sind Wörter über $(V \cup T)$
- P muss mindestens eine Variable enthalten
- Q ist beliebig

Bezeichnung:

P : Prämisse

Q : Conclusio

Schreibweise für Regeln

- Schreibweise für Regel (P, Q) :

$$P \rightarrow_G Q \quad \text{bzw.} \quad P \rightarrow Q$$

- Abkürzung für mehrere Regeln mit derselben Prämisse:

$$P \rightarrow Q_1 \mid Q_2 \mid Q_3 \quad \text{für} \quad P \rightarrow Q_1, P \rightarrow Q_2, P \rightarrow Q_3$$

Konvention (meistens)

- **Variablen** als Großbuchstaben
- **Terminale** als Kleinbuchstaben

Schreibweise für Regeln

- Schreibweise für Regel (P, Q) :

$$P \rightarrow_G Q \quad \text{bzw.} \quad P \rightarrow Q$$

- Abkürzung für mehrere Regeln mit derselben Prämisse:

$$P \rightarrow Q_1 \mid Q_2 \mid Q_3 \quad \text{für} \quad P \rightarrow Q_1, P \rightarrow Q_2, P \rightarrow Q_3$$

Konvention (meistens)

- **Variablen** als Großbuchstaben
- **Terminale** als Kleinbuchstaben

Schreibweise für Regeln

- Schreibweise für Regel (P, Q) :

$$P \rightarrow_G Q \quad \text{bzw.} \quad P \rightarrow Q$$

- Abkürzung für mehrere Regeln mit derselben Prämisse:

$$P \rightarrow Q_1 \mid Q_2 \mid Q_3 \quad \text{für} \quad P \rightarrow Q_1, P \rightarrow Q_2, P \rightarrow Q_3$$

Konvention (meistens)

- **Variablen** als Großbuchstaben
- **Terminale** als Kleinbuchstaben

Schreibweise für Regeln

- Schreibweise für Regel (P, Q) :

$$P \rightarrow_G Q \quad \text{bzw.} \quad P \rightarrow Q$$

- Abkürzung für mehrere Regeln mit derselben Prämisse:

$$P \rightarrow Q_1 \mid Q_2 \mid Q_3 \quad \text{für} \quad P \rightarrow Q_1, P \rightarrow Q_2, P \rightarrow Q_3$$

Konvention (meistens)

- **Variablen** als **Großbuchstaben**
- **Terminale** als **Kleinbuchstaben**

Beispiel 6.8

$S \rightarrow B$

$B \rightarrow \textit{do begin B end}$

$B \rightarrow A$

$A \rightarrow \textit{nop A}$

$A \rightarrow \varepsilon$

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

- 1 $aktuellWort := S$ (Startsymbol)
- 2 Wähle eine Regel $P \rightarrow Q$, so dass P in $aktuellWort$ vorkommt
- 3 Ersetze (ein) Vorkommen von P in $aktuellWort$ durch Q
- 4 Falls $aktuellWort$ noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort $aktuellWort$

Beachte

Die Berechnung

- ist nicht deterministisch (Auswahl der Regel)
- kann mehr als ein Ergebnis liefern (oder auch keines)
- kann in Endlosschleifen geraten

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

- 1 **aktuellWort** := S (Startsymbol)
- 2 Wähle eine Regel $P \rightarrow Q$, so dass P in *aktuellWort* vorkommt
- 3 Ersetze (ein) Vorkommen von P in *aktuellWort* durch Q
- 4 Falls *aktuellWort* noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort *aktuellWort*

Beachte

Die Berechnung

- ist nicht deterministisch (Auswahl der Regel)
- kann mehr als ein Ergebnis liefern (oder auch keines)
- kann in Endlosschleifen geraten

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

- 1 *aktuellWort* := S (Startsymbol)
- 2 Wähle eine Regel $P \rightarrow Q$, so dass P in *aktuellWort* vorkommt
- 3 Ersetze (ein) Vorkommen von P in *aktuellWort* durch Q
- 4 Falls *aktuellWort* noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort *aktuellWort*

Beachte

Die Berechnung

- ist nicht deterministisch (Auswahl der Regel)
- kann mehr als ein Ergebnis liefern (oder auch keines)
- kann in Endlosschleifen geraten

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

- 1 *aktuellWort* := S (Startsymbol)
- 2 Wähle eine Regel $P \rightarrow Q$, so dass P in *aktuellWort* vorkommt
- 3 **Ersetze (ein) Vorkommen von P in *aktuellWort* durch Q**
- 4 Falls *aktuellWort* noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort *aktuellWort*

Beachte

Die Berechnung

- ist nicht deterministisch (Auswahl der Regel)
- kann mehr als ein Ergebnis liefern (oder auch keines)
- kann in Endlosschleifen geraten

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

- 1 *aktuellWort* := S (Startsymbol)
- 2 Wähle eine Regel $P \rightarrow Q$, so dass P in *aktuellWort* vorkommt
- 3 Ersetze (ein) Vorkommen von P in *aktuellWort* durch Q
- 4 Falls *aktuellWort* noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort *aktuellWort*

Beachte

Die Berechnung

- ist nicht deterministisch (Auswahl der Regel)
- kann mehr als ein Ergebnis liefern (oder auch keines)
- kann in Endlosschleifen geraten

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

- 1 *aktuellWort* := S (Startsymbol)
- 2 Wähle eine Regel $P \rightarrow Q$, so dass P in *aktuellWort* vorkommt
- 3 Ersetze (ein) Vorkommen von P in *aktuellWort* durch Q
- 4 Falls *aktuellWort* noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort *aktuellWort*

Beachte

Die Berechnung

- ist nicht deterministisch (Auswahl der Regel)
- kann mehr als ein Ergebnis liefern (oder auch keines)
- kann in Endlosschleifen geraten

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

- 1 *aktuellWort* := S (Startsymbol)
- 2 Wähle eine Regel $P \rightarrow Q$, so dass P in *aktuellWort* vorkommt
- 3 Ersetze (ein) Vorkommen von P in *aktuellWort* durch Q
- 4 Falls *aktuellWort* noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort *aktuellWort*

Beachte

Die Berechnung

- ist nicht deterministisch (Auswahl der Regel)
- kann mehr als ein Ergebnis liefern (oder auch keines)
- kann in Endlosschleifen geraten

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

- 1 $aktuellWort := S$ (Startsymbol)
- 2 Wähle eine Regel $P \rightarrow Q$, so dass P in $aktuellWort$ vorkommt
- 3 Ersetze (ein) Vorkommen von P in $aktuellWort$ durch Q
- 4 Falls $aktuellWort$ noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort $aktuellWort$

Beachte

Die Berechnung

- **ist nicht deterministisch (Auswahl der Regel)**
- kann mehr als ein Ergebnis liefern (oder auch keines)
- kann in Endlosschleifen geraten

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

- 1 $aktuellWort := S$ (Startsymbol)
- 2 Wähle eine Regel $P \rightarrow Q$, so dass P in $aktuellWort$ vorkommt
- 3 Ersetze (ein) Vorkommen von P in $aktuellWort$ durch Q
- 4 Falls $aktuellWort$ noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort $aktuellWort$

Beachte

Die Berechnung

- ist nicht deterministisch (Auswahl der Regel)
- kann mehr als ein Ergebnis liefern (oder auch keines)
- kann in Endlosschleifen geraten

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

- 1 $aktuellWort := S$ (Startsymbol)
- 2 Wähle eine Regel $P \rightarrow Q$, so dass P in $aktuellWort$ vorkommt
- 3 Ersetze (ein) Vorkommen von P in $aktuellWort$ durch Q
- 4 Falls $aktuellWort$ noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort $aktuellWort$

Beachte

Die Berechnung

- ist nicht deterministisch (Auswahl der Regel)
- kann mehr als ein Ergebnis liefern (oder auch keines)
- **kann in Endlosschleifen geraten**

Beispiel 6.9 (Einfache Grammatiken)

Welche Wörter kann man ableiten?

- $G_a = (\{S\}, \{a\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow aS$$

$$R_2 = S \rightarrow \varepsilon$$

- $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow aSb$$

$$R_2 = S \rightarrow \varepsilon$$

- Sei $G_{gerade} = (\{S, S_0\}, \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow 1S \mid 2S_0 \mid 3S \mid 4S_0 \mid 5S \mid 6S_0 \mid 7S \mid 8S_0 \mid 9S$$

$$R_2 = S_0 \rightarrow S \mid \varepsilon$$

Beispiel 6.9 (Einfache Grammatiken)

Welche Wörter kann man ableiten?

- $G_a = (\{S\}, \{a\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow aS$$

$$R_2 = S \rightarrow \varepsilon$$

- $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow aSb$$

$$R_2 = S \rightarrow \varepsilon$$

- Sei $G_{gerade} = (\{S, S_0\}, \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow 1S \mid 2S_0 \mid 3S \mid 4S_0 \mid 5S \mid 6S_0 \mid 7S \mid 8S_0 \mid 9S$$

$$R_2 = S_0 \rightarrow S \mid \varepsilon$$

Beispiel 6.9 (Einfache Grammatiken)

Welche Wörter kann man ableiten?

- $G_a = (\{S\}, \{a\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow aS$$

$$R_2 = S \rightarrow \varepsilon$$

- $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow aSb$$

$$R_2 = S \rightarrow \varepsilon$$

- Sei $G_{gerade} = (\{S, S_0\}, \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, \{R_1, R_2\}, S)$

$$R_1 = S \rightarrow 1S \mid 2S_0 \mid 3S \mid 4S_0 \mid 5S \mid 6S_0 \mid 7S \mid 8S_0 \mid 9S$$

$$R_2 = S_0 \rightarrow S \mid \varepsilon$$

Definition 6.10 (Ableitung, Rechnung)

Gegeben:

- Grammatik $G = (V, T, R, S)$
- Wörter w, w' aus $(V \cup T)^*$

Es gilt

$$w \Longrightarrow_G w' \quad (\text{„}w \text{ geht über in } w'\text{“})$$

falls

$$\exists u, v \in (V \cup T)^* \exists P \rightarrow Q \in R \quad (w = uPv \text{ und } w' = uQv)$$

Definition 6.10 (Ableitung, Rechnung)

Gegeben:

- Grammatik $G = (V, T, R, S)$
- Wörter w, w' aus $(V \cup T)^*$

Es gilt

$$w \Longrightarrow_G w' \quad (\text{„}w \text{ geht über in } w'\text{“})$$

falls

$$\exists u, v \in (V \cup T)^* \exists P \rightarrow Q \in R \quad (w = uPv \text{ und } w' = uQv)$$

Schreibweise für Ableitung

$$w \Longrightarrow_G^* w'$$

falls es Wörter $w_0, \dots, w_n \in (V \cup T)^*$ ($n \geq 0$) gibt mit

- $w = w_0$
- $w_m = w'$
- $w_i \Longrightarrow_G w_{i+1}$ für $0 \leq i < n$

Merke: $w \Longrightarrow_G^* w$ gilt stets ($n = 0$)

Die Folge w_0, \dots, w_n heißt **Ableitung** oder **Rechnung**

- von w_0 nach w_n
- in G
- der Länge n

Schreibweise für Ableitung

$$w \Longrightarrow_G^* w'$$

falls es Wörter $w_0, \dots, w_n \in (V \cup T)^*$ ($n \geq 0$) gibt mit

- $w = w_0$
- $w_m = w'$
- $w_i \Longrightarrow_G w_{i+1}$ für $0 \leq i < n$

Merke: $w \Longrightarrow_G^* w$ gilt stets ($n = 0$)

Die Folge w_0, \dots, w_n heißt **Ableitung** oder **Rechnung**

- von w_0 nach w_n
- in G
- der Länge n

Schreibweise für Ableitung

$$w \Longrightarrow_G^* w'$$

falls es Wörter $w_0, \dots, w_n \in (V \cup T)^*$ ($n \geq 0$) gibt mit

- $w = w_0$
- $w_m = w'$
- $w_i \Longrightarrow_G w_{i+1}$ für $0 \leq i < n$

Merke: $w \Longrightarrow_G^* w$ gilt stets ($n = 0$)

Die Folge w_0, \dots, w_n heißt **Ableitung** oder **Rechnung**

- von w_0 nach w_n
- in G
- der Länge n

Beispiel 6.11 (Indeterminismus)

Wir betrachten die Grammatik $G = (\{S, B\}, \{a, b, c\}, \{R_0, R_1, R_2, R_3\}, S)$

$$R_0 = S \rightarrow aBBc$$

$$R_1 = B \rightarrow b$$

$$R_2 = B \rightarrow ba$$

$$R_3 = BB \rightarrow bBa$$

Drei Möglichkeiten, das Wort *abbac* zu erzeugen:

$$S \xRightarrow{R_0} aBBc \xRightarrow{R_1} abBc \xRightarrow{R_2} abbac$$

$$S \xRightarrow{R_0} aBBc \xRightarrow{R_2} aBbac \xRightarrow{R_1} abbac$$

$$S \xRightarrow{R_0} aBBc \xRightarrow{R_3} abBac \xRightarrow{R_1} abbac$$

Beispiel 6.11 (Indeterminismus)

Wir betrachten die Grammatik $G = (\{S, B\}, \{a, b, c\}, \{R_0, R_1, R_2, R_3\}, S)$

$$R_0 = S \rightarrow aBBc$$

$$R_1 = B \rightarrow b$$

$$R_2 = B \rightarrow ba$$

$$R_3 = BB \rightarrow bBa$$

Drei Möglichkeiten, das Wort *abbac* zu erzeugen:

$$S \xRightarrow{R_0} aBBc \xRightarrow{R_1} abBc \xRightarrow{R_2} abbac$$

$$S \xRightarrow{R_0} aBBc \xRightarrow{R_2} aBbac \xRightarrow{R_1} abbac$$

$$S \xRightarrow{R_0} aBBc \xRightarrow{R_3} abBac \xRightarrow{R_1} abbac$$

Warum ist das ein *Feature* und kein *Bug*?

- Erlaubt einfachere Definition von Grammatiken
- Für manche Sprachen gibt es keine eindeutige Grammatiken
- Eine Grammatik beschreibt die **Struktur** der Wörter.
Ein Wort kann mehrere mögliche Strukturen haben.
- Für **natürliche Sprachen** braucht man das unbedingt:
Manche Sätze sind mehrdeutig (in ihrer Grammatik),
also müssen auch die Grammatiken mehrdeutig sein!

Warum ist das ein *Feature* und kein *Bug*?

- Erlaubt einfachere Definition von Grammatiken
- Für manche Sprachen gibt es keine eindeutige Grammatiken
- Eine Grammatik beschreibt die Struktur der Wörter.
Ein Wort kann mehrere mögliche Strukturen haben.
- Für natürliche Sprachen braucht man das unbedingt:
Manche Sätze sind mehrdeutig (in ihrer Grammatik),
also müssen auch die Grammatiken mehrdeutig sein!

Warum ist das ein *Feature* und kein *Bug*?

- Erlaubt einfachere Definition von Grammatiken
- Für manche Sprachen gibt es keine eindeutige Grammatiken
- Eine Grammatik beschreibt die Struktur der Wörter.
Ein Wort kann mehrere mögliche Strukturen haben.
- Für natürliche Sprachen braucht man das unbedingt:
Manche Sätze sind mehrdeutig (in ihrer Grammatik),
also müssen auch die Grammatiken mehrdeutig sein!

Warum ist das ein *Feature* und kein *Bug*?

- Erlaubt einfachere Definition von Grammatiken
- Für manche Sprachen gibt es keine eindeutige Grammatiken
- Eine Grammatik beschreibt die **Struktur** der Wörter.
Ein Wort kann mehrere mögliche Strukturen haben.
- **Für natürliche Sprachen braucht man das unbedingt:
Manche Sätze sind mehrdeutig (in ihrer Grammatik),
also müssen auch die Grammatiken mehrdeutig sein!**

Beispiel 6.12 (Mehrdeutige Grammatik natürlichsprachlicher Sätze)

Time flies like an arrow.
Fruit flies like a banana.

- Beide Sätze haben zwei mögliche grammatische Strukturen.
- Erst unser semantisches Verständnis wählt eine aus.

Beispiel 6.12 (Mehrdeutige Grammatik natürlichsprachlicher Sätze)

Time flies like an arrow.
Fruit flies like a banana.

- Beide Sätze haben zwei mögliche grammatische Strukturen.
- Erst unser semantisches Verständnis wählt eine aus.

Erzeugte Sprache, Äquivalenz

Definition 6.13 (Erzeugte Sprache)

Gegeben: Eine Grammatik G

Die von G erzeugte Sprache $L(G)$ ist die Menge aller **terminalen** Wörter, die durch G vom Startsymbol S aus erzeugt werden können:

$$L(G) := \{w \in T^* \mid S \Longrightarrow_G^* w\}$$

Definition 6.14 (Äquivalenz)

Zwei Grammatiken G_1, G_2 heißen **äquivalent** gdw

$$L(G_1) = L(G_2)$$

Erzeugte Sprache, Äquivalenz

Definition 6.13 (Erzeugte Sprache)

Gegeben: Eine Grammatik G

Die von G erzeugte Sprache $L(G)$ ist die Menge aller **terminalen** Wörter, die durch G vom Startsymbol S aus erzeugt werden können:

$$L(G) := \{w \in T^* \mid S \Longrightarrow_G^* w\}$$

Definition 6.14 (Äquivalenz)

Zwei Grammatiken G_1, G_2 heißen **äquivalent** gdw

$$L(G_1) = L(G_2)$$

Beispiel

Beispiel 6.15

Grammatik $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$ mit

$$R_1 = S \rightarrow aSb$$

$$R_2 = S \rightarrow \varepsilon$$

Mögliche Ableitung:

$$S \xRightarrow{R_1} aSb \xRightarrow{R_1} aaSbb \xRightarrow{R_1} aaaSbbb \xRightarrow{R_2} aaabbb$$

Also: $a^3b^3 \in L(G_2)$

Lemma 6.16

Die Grammatik G_{ab} erzeugt die Sprache

$$L(G_{ab}) = \{a^n b^n \mid n \in \mathbb{N}_0\}$$

Beispiel

Beispiel 6.15

Grammatik $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$ mit

$$R_1 = S \rightarrow aSb$$

$$R_2 = S \rightarrow \varepsilon$$

Mögliche Ableitung:

$$S \xRightarrow{R_1} aSb \xRightarrow{R_1} aaSbb \xRightarrow{R_1} aaaSbbb \xRightarrow{R_2} aaabbbb$$

Also: $a^3b^3 \in L(G_2)$

Lemma 6.16

Die Grammatik G_{ab} erzeugt die Sprache

$$L(G_{ab}) = \{a^n b^n \mid n \in \mathbb{N}_0\}$$

Beispiel

Beispiel 6.15

Grammatik $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$ mit

$$R_1 = S \rightarrow aSb$$

$$R_2 = S \rightarrow \varepsilon$$

Mögliche Ableitung:

$$S \xRightarrow{R_1} aSb \xRightarrow{R_1} aaSbb \xRightarrow{R_1} aaaSbbb \xRightarrow{R_2} aaabbbb$$

Also: $a^3b^3 \in L(G_2)$

Lemma 6.16

Die Grammatik G_{ab} erzeugt die Sprache

$$L(G_{ab}) = \{a^n b^n \mid n \in \mathbb{N}_0\}$$

Beispiel

Beispiel 6.15

Grammatik $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$ mit

$$R_1 = S \rightarrow aSb$$

$$R_2 = S \rightarrow \varepsilon$$

Mögliche Ableitung:

$$S \xRightarrow{R_1} aSb \xRightarrow{R_1} aaSbb \xRightarrow{R_1} aaaSbbb \xRightarrow{R_2} aaabbbb$$

Also: $a^3b^3 \in L(G_2)$

Lemma 6.16

Die Grammatik G_{ab} erzeugt die Sprache

$$L(G_{ab}) = \{a^n b^n \mid n \in \mathbb{N}_0\}$$

Beweis

Dass G_{ab} tatsächlich genau diese Sprache erzeugt, zeigen wir allgemein, indem wir alle möglichen Ableitungen von G_{ab} betrachten.

\subseteq : zu zeigen: Jedes terminale Wort, das von G_{ab} erzeugt wird, hat die Form $a^n b^n$.

Wir zeigen für alle $w \in (V \cup T)^*$: Falls $S \xRightarrow{*}_{G_{ab}} w$, dann gilt entweder $w = a^n S b^n$ oder $w = a^n b^n$ für ein $n \in \mathbb{N}_0$.

Beweis

Dass G_{ab} tatsächlich genau diese Sprache erzeugt, zeigen wir allgemein, indem wir alle möglichen Ableitungen von G_{ab} betrachten.

\subseteq : zu zeigen: Jedes terminale Wort, das von G_{ab} erzeugt wird, hat die Form $a^n b^n$.

Wir zeigen für alle $w \in (V \cup T)^*$: Falls $S \xRightarrow{*}_{G_{ab}} w$, dann gilt entweder $w = a^n S b^n$ oder $w = a^n b^n$ für ein $n \in \mathbb{N}_0$.

Beweis (Forts.)

Dazu verwenden wir eine **Induktion über die Länge einer Ableitung** von S nach w .

Induktionsanfang: $w = S = a^0 S b^0$

Induktionsschritt: Es gelte $S \xRightarrow{*}_{G_{ab}} w \xRightarrow{G_{ab}} w'$, und für w gelte nach der Induktionsvoraussetzung bereits $w = a^n b^n$ oder $w = a^n S b^n$. Außerdem sei $w \xRightarrow{G_{ab}} w'$ eine Ableitung in einem Schritt. Nun ist zu zeigen: $w' = a^m b^m$ oder $w' = a^m S b^m$ für irgendein m .

Beweis (Forts.)

Dazu verwenden wir eine **Induktion über die Länge einer Ableitung** von S nach w .

Induktionsanfang: $w = S = a^0 S b^0$

Induktionsschritt: Es gelte $S \xRightarrow{*}_{G_{ab}} w \xRightarrow{G_{ab}} w'$, und für w gelte nach der Induktionsvoraussetzung bereits $w = a^n b^n$ oder $w = a^n S b^n$. Außerdem sei $w \xRightarrow{G_{ab}} w'$ eine Ableitung in einem Schritt. Nun ist zu zeigen: $w' = a^m b^m$ oder $w' = a^m S b^m$ für irgendein m .

Beweis (Forts.)

Dazu verwenden wir eine **Induktion über die Länge einer Ableitung** von S nach w .

Induktionsanfang: $w = S = a^0 S b^0$

Induktionsschritt: Es gelte $S \xRightarrow{*}_{G_{ab}} w \xRightarrow{G_{ab}} w'$, und für w gelte nach der Induktionsvoraussetzung bereits $w = a^n b^n$ oder $w = a^n S b^n$. Außerdem sei $w \xRightarrow{G_{ab}} w'$ eine Ableitung in einem Schritt. Nun ist zu zeigen: $w' = a^m b^m$ oder $w' = a^m S b^m$ für irgendein m .

Beweis (Forts.)

Fall 1: $w = a^n b^n$. Dann konnte keine Regel angewandt werden, da w schon terminal ist, also tritt dieser Fall nie auf.

Fall 2: $w = a^n S b^n$. Dann wurde von w nach w' entweder Regel R_1 oder R_2 angewandt.

Falls R_1 angewandt wurde, dann gilt $w = a^n S b^n \xRightarrow{R_1} a^n a S b b^n = a^{n+1} S b^{n+1} = w'$.

Falls R_2 angewandt wurde, dann gilt $w = a^n S b^n \xRightarrow{R_2} a^n \epsilon b^n = w'$. Dies Wort ist terminal und hat die geforderte Form $a^n b^n$.

Beweis (Forts.)

Fall 1: $w = a^n b^n$. Dann konnte keine Regel angewandt werden, da w schon terminal ist, also tritt dieser Fall nie auf.

Fall 2: $w = a^n S b^n$. Dann wurde von w nach w' entweder Regel R_1 oder R_2 angewandt.

Falls R_1 angewandt wurde, dann gilt $w = a^n S b^n \xRightarrow{R_1} a^n a S b b^n = a^{n+1} S b^{n+1} = w'$.

Falls R_2 angewandt wurde, dann gilt $w = a^n S b^n \xRightarrow{R_2} a^n \epsilon b^n = w'$. Dies Wort ist terminal und hat die geforderte Form $a^n b^n$.

Beweis (Forts.)

\supseteq : zu zeigen: Für alle n kann $a^n b^n$ von G_{ab} erzeugt werden: $S \xRightarrow{*}_{G_{ab}} a^n b^n$
 $\forall n \in \mathbb{N}_0$.

Um $a^n b^n$ zu erzeugen, wende man auf S n -mal die Regel R_1 und dann einmal die Regel R_2 an. □

Beispiel: Dycksprache

Definition 6.17 (Dycksprache)

Gegeben:

- $k \in \mathbb{N}$
- $\Sigma_k := \{x_1, \bar{x}_1, x_2, \dots, x_k, \bar{x}_k\}$ ein Alphabet mit $2k$ Symbolen

Die Dycksprache D_k ist die kleinste Menge die folgende Bedingungen erfüllt:

- 1 $\varepsilon \in D_k$,
- 2 Falls $w \in D_k$, so auch $x_i w \bar{x}_i$.
- 3 Falls $u, v \in D_k$, so auch uv .

Interpretiert man die x_i als öffnende, die \bar{x}_i als zugehörige schließende Klammern, so kann man die Dycksprache als die Menge aller korrekten Klammerausdrücke sehen.

Beispiel: Dycksprache

Definition 6.17 (Dycksprache)

Gegeben:

- $k \in \mathbb{N}$
- $\Sigma_k := \{x_1, \bar{x}_1, x_2, \dots, x_k, \bar{x}_k\}$ ein Alphabet mit $2k$ Symbolen

Die Dycksprache D_k ist die **kleinste Menge** die folgende Bedingungen erfüllt:

- 1 $\varepsilon \in D_k$,
- 2 Falls $w \in D_k$, so auch $x_i w \bar{x}_i$.
- 3 Falls $u, v \in D_k$, so auch uv .

Interpretiert man die x_i als öffnende, die \bar{x}_i als zugehörige schließende Klammern, so kann man die Dycksprache als die **Menge aller korrekten Klammerausdrücke** sehen.

Beispiel: Dycksprache

Definition 6.17 (Dycksprache)

Gegeben:

- $k \in \mathbb{N}$
- $\Sigma_k := \{x_1, \bar{x}_1, x_2, \dots, x_k, \bar{x}_k\}$ ein Alphabet mit $2k$ Symbolen

Die Dycksprache D_k ist die **kleinste Menge** die folgende Bedingungen erfüllt:

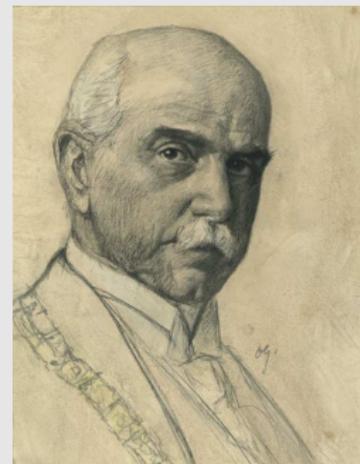
- 1 $\varepsilon \in D_k$,
- 2 Falls $w \in D_k$, so auch $x_i w \bar{x}_i$.
- 3 Falls $u, v \in D_k$, so auch uv .

Interpretiert man die x_i als öffnende, die \bar{x}_i als zugehörige schließende Klammern, so kann man die Dycksprache als die **Menge aller korrekten Klammerausdrücke** sehen.

Beispiel: Dycksprache

Walther von Dyck ★ 1856, † 1934

- Mathematiker
- Hochschulpolitiker
- Erster Rektor der TU München
- Einer der Gründungsväter des Deutschen Museums



[Foto: Deutsches Museum]

Teil II

Terminologie

- 1 **Sprache, Grammatik**
- 2 Warum Sprachen?
- 3 Die Chomsky-Hierarchie
- 4 Probleme über Sprachen
- 5 Endlich, unendlich und dann?

Teil II

Terminologie

- 1 Sprache, Grammatik
- 2 Warum Sprachen?**
- 3 Die Chomsky-Hierarchie
- 4 Probleme über Sprachen
- 5 Endlich, unendlich und dann?

Fakt

So ziemlich alle Probleme können als Probleme über Sprachen formuliert werden.

Beispiel 7.1 (Primzahlen)

Alphabet $\Sigma_{\text{num}} := \{\mid\}$

Sprache $L_{\text{primes}} := \{\underbrace{\mid \dots \mid}_{p \text{ mal}} \mid p \text{ prim}\}$

Fakt

So ziemlich alle Probleme können als Probleme über Sprachen formuliert werden.

Beispiel 7.1 (Primzahlen)

Alphabet $\Sigma_{\text{num}} := \{|\}$

Sprache $L_{\text{primes}} := \{ \underbrace{|\dots|}_{p \text{ mal}} \mid p \text{ prim} \}$

Eingabealphabet

$$\Sigma = \{0, 1, \dots, n-1\}$$

erlaubt Darstellung einer Ganzzahl zur Basis n

Beispiel 7.2

5 binär: 101

5 unär: ||||| (oder auch 11111)

Eingabealphabet

$$\Sigma = \{0, 1, \dots, n-1\}$$

erlaubt Darstellung einer Ganzzahl zur Basis n

Beispiel 7.2

5 binär: 101

5 unär: |||| (oder auch 11111)

Speicheraufwand

n -äre Darstellung ($n > 1$) einer Zahl k führt zu einer Speicherersparnis:

$\log_n k$ (n -är) statt k (unär)

Nur der Schritt von unär auf binär ist wesentlich, denn

$$\log_n k = \frac{1}{\log_2 n} \cdot \log_2 k = c \cdot \log_2 k$$

(von binär auf n -är nur lineare Einsparung)

Speicheraufwand

n -äre Darstellung ($n > 1$) einer Zahl k führt zu einer Speicherersparnis:

$\log_n k$ (n -är) statt k (unär)

Nur der Schritt von unär auf binär ist wesentlich, denn

$$\log_n k = \frac{1}{\log_2 n} \cdot \log_2 k = c \cdot \log_2 k$$

(von binär auf n -är nur lineare Einsparung)

Darstellung des Erfüllbarkeitsproblems SAT

Problem SAT

Gegeben: Eine aussagenlogische Formel w

Frage: Gibt es eine Belegung der booleschen Variablen in w , so dass w zu *true* auswertet?

Signatur für aussagenlogische Formeln

Signatur: $\Sigma_{\text{sat}} := \{\wedge, \vee, \neg, (,), x, 0, 1\}$

Dabei Darstellung von boolescher Variablen x_i als x gefolgt von i binär kodiert. Dadurch Formel der Länge n um (unerheblichen) Faktor $\log n$ länger.

Darstellung des Erfüllbarkeitsproblems SAT

Problem SAT

Gegeben: Eine aussagenlogische Formel w

Frage: Gibt es eine Belegung der booleschen Variablen in w , so dass w zu *true* auswertet?

Signatur für aussagenlogische Formeln

Signatur: $\Sigma_{\text{sat}} := \{\wedge, \vee, \neg, (,), x, 0, 1\}$

Dabei Darstellung von boolescher Variablen x_i als x gefolgt von i binär kodiert. Dadurch Formel der Länge n um (unerheblichen) Faktor $\log n$ länger.

Definition 7.3 (Satisfiability)

Sprache

$L_{\text{sat}} := \{ w \in \Sigma_{\text{sat}}^* : w \text{ ist eine aussagenlogische Formel,} \\ \text{und es gibt eine Belegung für die } x_i, \\ \text{so dass die Formel } w \text{ zu } \textit{true} \text{ auswertet} \}$

Darstellung des Erreichbarkeitsproblems in Graphen

Erreichbarkeitsproblem

Gegeben: Ein Graph mit Ecken v_1 bis v_n

Frage: Gibt es einen Weg von Ecke v_1 zu Ecke v_n ?

Signatur für Graphen

Signatur: $\Sigma_{\text{graph}} := \{v, e, 0, 1, (,), \#\}$

Darstellung von

Ecke v_i als v gefolgt i binär kodiert

Kante $e_{i,j}$ als $e(\text{string}_1\#\text{string}_2)$, wobei

- string_1 die binäre Darstellung von i ,
- string_2 die binäre Darstellung von j

Darstellung des Erreichbarkeitsproblems in Graphen

Erreichbarkeitsproblem

Gegeben: Ein Graph mit Ecken v_1 bis v_n

Frage: Gibt es einen Weg von Ecke v_1 zu Ecke v_n ?

Signatur für Graphen

Signatur: $\Sigma_{\text{graph}} := \{v, e, 0, 1, (,), \#\}$

Darstellung von

Ecke v_i als v gefolgt i binär kodiert

Kante $e_{i,j}$ als $e(\text{string}_1\#\text{string}_2)$, wobei

- string_1 die binäre Darstellung von i ,
- string_2 die binäre Darstellung von j

Darstellung des Erreichbarkeitsproblems in Graphen

Erreichbarkeitsproblem

Gegeben: Ein Graph mit Ecken v_1 bis v_n

Frage: Gibt es einen Weg von Ecke v_1 zu Ecke v_n ?

Signatur für Graphen

Signatur: $\Sigma_{\text{graph}} := \{v, e, 0, 1, (,), \#\}$

Darstellung von

Ecke v_i als v gefolgt i binär kodiert

Kante $e_{i,j}$ als $e(\text{string}_1\#\text{string}_2)$, wobei

- string_1 die binäre Darstellung von i ,
- string_2 die binäre Darstellung von j

Definition 7.4 (Erreichbarkeitsproblem)

Sprache

$L_{\text{reach}} := \{w \in \Sigma_{\text{graph}}^* : \text{es gibt einen Weg in } w \text{ von der ersten Ecke } v_1 \text{ zur letzten Ecke } v_n\}$

Teil II

Terminologie

- 1 Sprache, Grammatik
- 2 Warum Sprachen?**
- 3 Die Chomsky-Hierarchie
- 4 Probleme über Sprachen
- 5 Endlich, unendlich und dann?

Teil II

Terminologie

- 1 Sprache, Grammatik
- 2 Warum Sprachen?
- 3 Die Chomsky-Hierarchie**
- 4 Probleme über Sprachen
- 5 Endlich, unendlich und dann?

Die Chomsky-Hierarchie

Noam Chomsky ★ 1928

- Professor für Linguistik und Philosophie am MIT
- Bedeutender Linguist
- Bedeutender Beitrag zur Informatik:
Erste Beschreibung der Chomsky-Hierarchie
(1956)
- Bedeutender linker Intellektueller und
Globalisierungskritiker



Die Chomsky-Hierarchie

Was muss eine Grammatik erfüllen?

- Sie darf nur **endlich viele Regeln** haben
 - Jede Regelprämisse muss **mindestens eine Variable** enthalten

Das Wort kann im Lauf der Ableitung beliebig wachsen und wieder schrumpfen.

(Weitere) Beschränkung der Form, die Regeln haben dürfen, führt zu

- **Grammatiktypen** und damit auch zu
- **Sprachtypen**

von verschiedenen Schwierigkeitsgraden.

Die Chomsky-Hierarchie

Was muss eine Grammatik erfüllen?

- Sie darf nur **endlich viele Regeln** haben
- **Jede Regelprämisse muss mindestens eine Variable enthalten**

Das Wort kann im Lauf der Ableitung beliebig wachsen und wieder schrumpfen.

(Weitere) Beschränkung der Form, die Regeln haben dürfen, führt zu

- **Grammatiktypen** und damit auch zu
- **Sprachtypen**

von verschiedenen Schwierigkeitsgraden.

Die Chomsky-Hierarchie

Was muss eine Grammatik erfüllen?

- Sie darf nur **endlich viele Regeln** haben
- Jede Regelprämissse muss **mindestens eine Variable** enthalten

Das Wort kann im Lauf der Ableitung beliebig wachsen und wieder schrumpfen.

(Weitere) Beschränkung der Form, die Regeln haben dürfen, führt zu

- **Grammatiktypen** und damit auch zu
- **Sprachtypen**

von verschiedenen Schwierigkeitsgraden.

Die Chomsky-Hierarchie

Was muss eine Grammatik erfüllen?

- Sie darf nur **endlich viele Regeln** haben
- Jede Regelprämisse muss **mindestens eine Variable** enthalten

Das Wort kann im Lauf der Ableitung beliebig wachsen und wieder schrumpfen.

(Weitere) Beschränkung der Form, die Regeln haben dürfen, führt zu

- **Grammatiktypen** und damit auch zu
- **Sprachtypen**

von verschiedenen Schwierigkeitsgraden.

Die Chomsky-Hierarchie

Was muss eine Grammatik erfüllen?

- Sie darf nur **endlich viele Regeln** haben
- Jede Regelprämisse muss **mindestens eine Variable** enthalten

Das Wort kann im Lauf der Ableitung beliebig wachsen und wieder schrumpfen.

(Weitere) Beschränkung der Form, die Regeln haben dürfen, führt zu

- **Grammatiktypen** und damit auch zu
- **Sprachtypen**

von verschiedenen Schwierigkeitsgraden.

Definition 8.1 (Rechtlineare Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **rechtslinear** gdw

$$\forall (P \rightarrow Q) \in R \quad (P \in V \text{ und } Q \in T^* \cup T^+ V)$$

Das heißt, bei jeder Regelanwendung:

- Links eine **einzelne Variable**
- Rechts **höchstens eine Variable**
- Wenn rechts eine Variable steht, steht sie **ganz rechts im Wort**.

Definition 8.1 (Rechtlineare Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **rechtslinear** gdw

$$\forall (P \rightarrow Q) \in R \quad (P \in V \text{ und } Q \in T^* \cup T^+ V)$$

Das heißt, bei jeder Regelanwendung:

- Links eine **einzelne Variable**
- Rechts **höchstens eine Variable**
- Wenn rechts eine Variable steht, steht sie **ganz rechts im Wort**.

Definition 8.1 (Rechtlineare Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **rechtslinear** gdw

$$\forall (P \rightarrow Q) \in R \quad (P \in V \text{ und } Q \in T^* \cup T^+ V)$$

Das heißt, bei jeder Regelanwendung:

- Links eine **einzelne Variable**
- **Rechts höchstens eine Variable**
- Wenn rechts eine Variable steht, steht sie **ganz rechts im Wort**.

Definition 8.1 (Rechtlineare Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **rechtslinear** gdw

$$\forall (P \rightarrow Q) \in R \quad (P \in V \text{ und } Q \in T^* \cup T^+ V)$$

Das heißt, bei jeder Regelanwendung:

- Links eine **einzelne Variable**
- Rechts **höchstens eine Variable**
- **Wenn rechts eine Variable steht, steht sie ganz rechts im Wort.**

Die Chomsky-Hierarchie

Definition 8.2 (Kontextfreie Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextfrei** gdw

$$\forall (P \rightarrow Q) \in R \quad (P \in V \text{ und } Q \in (V \cup T)^*)$$

Das heißt, bei jeder Regelanwendung:

- Links eine **einzelne Variable**
- Die Prämisse macht keine Aussage, was der Kontext dieser Variablen ist („kontextfrei“)
- Rechts steht etwas beliebiges

Definition 8.2 (Kontextfreie Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextfrei** gdw

$$\forall (P \rightarrow Q) \in R \quad (P \in V \text{ und } Q \in (V \cup T)^*)$$

Das heißt, bei jeder Regelanwendung:

- Links eine **einzelne Variable**
- Die Prämisse macht keine Aussage, was der Kontext dieser Variablen ist („kontextfrei“)
- Rechts steht etwas beliebiges

Die Chomsky-Hierarchie

Definition 8.2 (Kontextfreie Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextfrei** gdw

$$\forall (P \rightarrow Q) \in R \quad (P \in V \text{ und } Q \in (V \cup T)^*)$$

Das heißt, bei jeder Regelanwendung:

- Links eine **einzelne Variable**
- Die Prämisse macht keine Aussage, was der Kontext dieser Variablen ist („kontextfrei“)
- Rechts steht etwas beliebiges

Definition 8.2 (Kontextfreie Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextfrei** gdw

$$\forall (P \rightarrow Q) \in R \quad (P \in V \text{ und } Q \in (V \cup T)^*)$$

Das heißt, bei jeder Regelanwendung:

- Links eine **einzelne Variable**
- Die Prämisse macht keine Aussage, was der Kontext dieser Variablen ist („kontextfrei“)
- **Rechts steht etwas beliebiges**

Die Chomsky-Hierarchie

Definition 8.3 (Kontextsensitive Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextsensitiv** gdw

$\forall (P \rightarrow Q) \in R:$

- 1 $\exists u, v, \alpha \in (V \cup T)^* \exists A \in V (P = uAv \text{ und } Q = u\alpha v \text{ mit } |\alpha| \geq 1)$, **oder**
die Regel hat die Form $S \rightarrow \varepsilon$
- 2 S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Eine Variable A wird in einen String α mit $|\alpha| \geq 1$ überführt
- Die Ersetzung von A durch α findet nur statt, wenn der in der Regel geforderte **Kontext** (u und v), vorhanden ist
- Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$

Die Chomsky-Hierarchie

Definition 8.3 (Kontextsensitive Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextsensitiv** gdw

$\forall (P \rightarrow Q) \in R:$

- 1 $\exists u, v, \alpha \in (V \cup T)^* \exists A \in V (P = uAv \text{ und } Q = u\alpha v \text{ mit } |\alpha| \geq 1)$, **oder**
die Regel hat die Form $S \rightarrow \varepsilon$
- 2 S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Eine Variable A wird in einen String α mit $|\alpha| \geq 1$ überführt
- Die Ersetzung von A durch α findet nur statt, wenn der in der Regel geforderte **Kontext** (u und v), vorhanden ist
- Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$

Die Chomsky-Hierarchie

Definition 8.3 (Kontextsensitive Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextsensitiv** gdw

$\forall (P \rightarrow Q) \in R:$

- 1 $\exists u, v, \alpha \in (V \cup T)^* \exists A \in V (P = uAv \text{ und } Q = u\alpha v \text{ mit } |\alpha| \geq 1)$, **oder**
die Regel hat die Form $S \rightarrow \varepsilon$
- 2 S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Eine Variable A wird in einen String α mit $|\alpha| \geq 1$ überführt
- Die Ersetzung von A durch α findet nur statt, wenn der in der Regel geforderte **Kontext** (u und v), vorhanden ist
- Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$

Die Chomsky-Hierarchie

Definition 8.3 (Kontextsensitive Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextsensitiv** gdw

$\forall (P \rightarrow Q) \in R:$

- 1 $\exists u, v, \alpha \in (V \cup T)^* \exists A \in V (P = uAv \text{ und } Q = u\alpha v \text{ mit } |\alpha| \geq 1)$, **oder**
die Regel hat die Form $S \rightarrow \varepsilon$
- 2 S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Eine Variable A wird in einen String α mit $|\alpha| \geq 1$ überführt
- Die Ersetzung von A durch α findet nur statt, wenn der in der Regel geforderte **Kontext** (u und v), vorhanden ist
- Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$

Die Chomsky-Hierarchie

Definition 8.3 (Kontextsensitive Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextsensitiv** gdw

$\forall (P \rightarrow Q) \in R:$

- 1 $\exists u, v, \alpha \in (V \cup T)^* \exists A \in V (P = uAv \text{ und } Q = u\alpha v \text{ mit } |\alpha| \geq 1)$, **oder**
die Regel hat die Form $S \rightarrow \varepsilon$
- 2 S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Eine Variable A wird in einen String α mit $|\alpha| \geq 1$ überführt
- Die Ersetzung von A durch α findet nur statt, wenn der in der Regel geforderte **Kontext** (u und v), vorhanden ist
- Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$

Definition 8.3 (Kontextsensitive Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextsensitiv** gdw

$\forall (P \rightarrow Q) \in R:$

- 1 $\exists u, v, \alpha \in (V \cup T)^* \exists A \in V (P = uAv \text{ und } Q = u\alpha v \text{ mit } |\alpha| \geq 1)$, **oder**
die Regel hat die Form $S \rightarrow \varepsilon$
- 2 S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Eine Variable A wird in einen String α mit $|\alpha| \geq 1$ überführt
- Die Ersetzung von A durch α findet nur statt, wenn der in der Regel geforderte **Kontext** (u und v), vorhanden ist
- **Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$**

Die Chomsky-Hierarchie

Definition 8.4 (Beschränkte Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **beschränkt** gdw

$\forall (P \rightarrow Q) \in R:$

① $|P| \leq |Q|$, **oder**

die Regel hat die Form $S \rightarrow \varepsilon$

② S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Die Conclusio ist mindestens so lang wie die Prämisse, außer bei $\varepsilon \in L$.
- Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$.

Definition 8.4 (Beschränkte Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **beschränkt** gdw

$\forall (P \rightarrow Q) \in R:$

① $|P| \leq |Q|$, **oder**

die Regel hat die Form $S \rightarrow \varepsilon$

② S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Die Conclusio ist mindestens so lang wie die Prämisse, außer bei $\varepsilon \in L$.
- Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$.

Definition 8.4 (Beschränkte Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **beschränkt** gdw

$\forall (P \rightarrow Q) \in R:$

① $|P| \leq |Q|$, **oder**

die Regel hat die Form $S \rightarrow \varepsilon$

② S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Die Conclusio ist mindestens so lang wie die Prämisse, außer bei $\varepsilon \in L$.
- Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$.

Definition 8.4 (Beschränkte Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **beschränkt** gdw

$\forall (P \rightarrow Q) \in R:$

① $|P| \leq |Q|$, **oder**

die Regel hat die Form $S \rightarrow \varepsilon$

② S nicht in Q

Das heißt, bei jeder Regelanwendung:

● Die **Conclusio** ist mindestens so lang wie die Prämisse, außer bei $\varepsilon \in L$.

● Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$

Definition 8.4 (Beschränkte Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **beschränkt** gdw

$\forall (P \rightarrow Q) \in R:$

- 1 $|P| \leq |Q|$, **oder**

die Regel hat die Form $S \rightarrow \varepsilon$

- 2 S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Die Conclusio ist mindestens so lang wie die Prämisse, außer bei $\varepsilon \in L$.
- **Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$**

Die Chomsky-Hierarchie

Aufbauend auf den Grammatikarten kann man Sprachklassen definieren

Definition 8.5 (Sprachklassen)

Klasse	definiert als	Sprache heißt
L_3 , REG	$\{L(G) \mid G \text{ ist rechtslinear}\}$	Typ 3, regulär
L_2 , CFL	$\{L(G) \mid G \text{ ist kontextfrei}\}$	Typ 2, kontextfrei
L_1 , CSL	$\{L(G) \mid G \text{ ist kontextsensitiv}\}$	Typ 1, kontextsensitiv
L_1 , CSL	$\{L(G) \mid G \text{ ist beschränkt}\}$	Typ 1, beschränkt
L_0 , r.e.	$\{L(G) \mid G \text{ beliebig}\}$	Typ 0, aufzählbar
L	$\{L \mid L \subseteq \Sigma^*\}$	beliebige Sprache

Die Chomsky-Hierarchie

Aufbauend auf den Grammatikarten kann man Sprachklassen definieren

Definition 8.5 (Sprachklassen)

Klasse	definiert als	Sprache heißt
L_3 , REG	$\{L(G) \mid G \text{ ist rechtslinear}\}$	Typ 3, regulär
L_2 , CFL	$\{L(G) \mid G \text{ ist kontextfrei}\}$	Typ 2, kontextfrei
L_1 , CSL	$\{L(G) \mid G \text{ ist kontextsensitiv}\}$	Typ 1, kontextsensitiv
L_1 , CSL	$\{L(G) \mid G \text{ ist beschränkt}\}$	Typ 1, beschränkt
L_0 , r.e.	$\{L(G) \mid G \text{ beliebig}\}$	Typ 0, aufzählbar
L	$\{L \mid L \subseteq \Sigma^*\}$	beliebige Sprache

Die Chomsky-Hierarchie

Grammatiken können kompliziert sein!

Beispiel 8.6 ()

Grammatik $G_{abc} = (\{S, X_1, X_2\}, \{a, b, c\}, \{R_1, \dots, R_5\}, S)$ mit

$$R_1 = S \rightarrow abc \mid aX_1bc$$

$$R_2 = X_1b \rightarrow bX_1$$

$$R_3 = X_1c \rightarrow X_2bcc$$

$$R_4 = bX_2 \rightarrow X_2b$$

$$R_5 = aX_2 \rightarrow aa \mid aaX_1$$

- Ist diese Grammatik **kontextsensitiv**?
- Ist sie **beschränkt**?

Die Chomsky-Hierarchie

Grammatiken können kompliziert sein!

Beispiel 8.6 ()

Grammatik $G_{abc} = (\{S, X_1, X_2\}, \{a, b, c\}, \{R_1, \dots, R_5\}, S)$ mit

$$R_1 = S \rightarrow abc \mid aX_1bc$$

$$R_2 = X_1b \rightarrow bX_1$$

$$R_3 = X_1c \rightarrow X_2bcc$$

$$R_4 = bX_2 \rightarrow X_2b$$

$$R_5 = aX_2 \rightarrow aa \mid aaX_1$$

- Ist diese Grammatik **kontextsensitiv**?
- Ist sie **beschränkt**?

Die Chomsky-Hierarchie

Grammatiken können kompliziert sein!

Beispiel 8.6 ()

Grammatik $G_{abc} = (\{S, X_1, X_2\}, \{a, b, c\}, \{R_1, \dots, R_5\}, S)$ mit

$$R_1 = S \rightarrow abc \mid aX_1bc$$

$$R_2 = X_1b \rightarrow bX_1$$

$$R_3 = X_1c \rightarrow X_2bcc$$

$$R_4 = bX_2 \rightarrow X_2b$$

$$R_5 = aX_2 \rightarrow aa \mid aaX_1$$

- Ist diese Grammatik **kontextsensitiv**?
- Ist sie **beschränkt**?

Die Chomsky-Hierarchie

Grammatiken können kompliziert sein!

Beispiel 8.6 ()

Grammatik $G_{abc} = (\{S, X_1, X_2\}, \{a, b, c\}, \{R_1, \dots, R_5\}, S)$ mit

$$R_1 = S \rightarrow abc \mid aX_1bc$$

$$R_2 = X_1b \rightarrow bX_1$$

$$R_3 = X_1c \rightarrow X_2bcc$$

$$R_4 = bX_2 \rightarrow X_2b$$

$$R_5 = aX_2 \rightarrow aa \mid aaX_1$$

- Ist diese Grammatik **kontextsensitiv**?
- Ist sie **beschränkt**?

Die Chomsky-Hierarchie

Grammatiken können kompliziert sein!

Beispiel 8.6 (Grammatik für $a^n b^n c^n$)

Grammatik $G_{abc} = (\{S, X_1, X_2\}, \{a, b, c\}, \{R_1, \dots, R_5\}, S)$ mit

$$R_1 = S \quad \rightarrow \quad abc \mid aX_1bc$$

$$R_2 = X_1b \quad \rightarrow \quad bX_1$$

$$R_3 = X_1c \quad \rightarrow \quad X_2bcc$$

$$R_4 = bX_2 \quad \rightarrow \quad X_2b$$

$$R_5 = aX_2 \quad \rightarrow \quad aa \mid aaX_1$$

- Ist diese Grammatik **kontextsensitiv**?
- Ist sie **beschränkt**?

Teil II

Terminologie

- 1 Sprache, Grammatik
- 2 Warum Sprachen?
- 3 Die Chomsky-Hierarchie**
- 4 Probleme über Sprachen
- 5 Endlich, unendlich und dann?

Teil II

Terminologie

- 1 Sprache, Grammatik
- 2 Warum Sprachen?
- 3 Die Chomsky-Hierarchie
- 4 Probleme über Sprachen**
- 5 Endlich, unendlich und dann?

Interessante Probleme (informell)

- Ist ein gegebenes Wort in einer Sprache (definiert durch eine Grammatik) enthalten?
- Erzeugen zwei gegebene Grammatiken dieselbe Sprache?

Mit welchen Algorithmen können diese Probleme gelöst werden?

Interessante Probleme (informell)

- Ist ein gegebenes Wort in einer Sprache (definiert durch eine Grammatik) enthalten?
- Erzeugen zwei gegebene Grammatiken dieselbe Sprache?

Mit welchen Algorithmen können diese Probleme gelöst werden?

Interessante Probleme (informell)

- Ist ein gegebenes Wort in einer Sprache (definiert durch eine Grammatik) enthalten?
- Erzeugen zwei gegebene Grammatiken dieselbe Sprache?

Mit welchen Algorithmen können diese Probleme gelöst werden?

Interessante Probleme (informell)

- Ist ein gegebenes Wort in einer Sprache (definiert durch eine Grammatik) enthalten?
- Erzeugen zwei gegebene Grammatiken dieselbe Sprache?

Mit welchen Algorithmen können diese Probleme gelöst werden?

Definition 9.1 (Problem, Algorithmus)

Ein **Problem** P ist die Frage, ob eine bestimmte Eigenschaft auf gegebene Objekte zutrifft.

Dabei ist eine bekannte, abzählbaren Grundmenge solcher Objekte gegeben.

Für jedes Objekt o gilt: die Eigenschaft trifft auf o zu oder nicht.

Definition 9.2 (Algorithmus)

Ein **Algorithmus** für ein Problem P ist eine Vorschrift (ein Programm), die zu beliebigem Objekt o berechnet, ob die Eigenschaft für o zutrifft oder nicht.

Definition 9.1 (Problem, Algorithmus)

Ein **Problem** P ist die Frage, ob eine bestimmte Eigenschaft auf gegebene Objekte zutrifft.

Dabei ist eine bekannte, abzählbaren Grundmenge solcher Objekte gegeben.

Für jedes Objekt o gilt: die Eigenschaft trifft auf o zu oder nicht.

Definition 9.2 (Algorithmus)

Ein **Algorithmus** für ein Problem P ist eine Vorschrift (ein Programm), die zu beliebigem Objekt o berechnet, ob die Eigenschaft für o zutrifft oder nicht.

Definition 9.1 (Problem, Algorithmus)

Ein **Problem** P ist die Frage, ob eine bestimmte Eigenschaft auf gegebene Objekte zutrifft.

Dabei ist eine bekannte, abzählbaren Grundmenge solcher Objekte gegeben.

Für jedes Objekt o gilt: die Eigenschaft trifft auf o zu oder nicht.

Definition 9.2 (Algorithmus)

Ein **Algorithmus** für ein Problem P ist eine Vorschrift (ein Programm), die zu beliebigem Objekt o berechnet, ob die Eigenschaft für o zutrifft oder nicht.

Definition 9.1 (Problem, Algorithmus)

Ein **Problem** P ist die Frage, ob eine bestimmte Eigenschaft auf gegebene Objekte zutrifft.

Dabei ist eine bekannte, abzählbaren Grundmenge solcher Objekte gegeben.

Für jedes Objekt o gilt: die Eigenschaft trifft auf o zu oder nicht.

Definition 9.2 (Algorithmus)

Ein **Algorithmus** für ein Problem P ist eine Vorschrift (ein Programm), die zu beliebigem Objekt o berechnet, ob die Eigenschaft für o zutrifft oder nicht.

Beispiel 9.3 (Einige Probleme)

- Für $n \in \mathbb{N}$:
Ist n eine Primzahl?
- Für ein Wort $w \in \Sigma^*$ und
ein Element G aus der Menge aller Grammatiken über Σ :
Gilt $w \in L(G)$?
- Für ein Element G aus der Menge aller Grammatiken:
Ist $L(G)$ leer (endlich, unendlich)?
- Für $(a, b, c) \in \mathbb{N}^3$:
Hat $a^n + b^n = c^n$ eine Lösung in den natürlichen Zahlen?
- Für ein Programm p aus der Menge aller Java-Programme:
Terminiert p ?

Beispiel 9.3 (Einige Probleme)

- Für $n \in \mathbb{N}$:
Ist n eine Primzahl?
- Für ein Wort $w \in \Sigma^*$ und
ein Element G aus der Menge aller Grammatiken über Σ :
Gilt $w \in L(G)$?
- Für ein Element G aus der Menge aller Grammatiken:
Ist $L(G)$ leer (endlich, unendlich)?
- Für $(a, b, c) \in \mathbb{N}^3$:
Hat $a^n + b^n = c^n$ eine Lösung in den natürlichen Zahlen?
- Für ein Programm p aus der Menge aller Java-Programme:
Terminiert p ?

Beispiel 9.3 (Einige Probleme)

- Für $n \in \mathbb{N}$:
Ist n eine Primzahl?
- Für ein Wort $w \in \Sigma^*$ und
ein Element G aus der Menge aller Grammatiken über Σ :
Gilt $w \in L(G)$?
- Für ein Element G aus der Menge aller Grammatiken:
Ist $L(G)$ leer (endlich, unendlich)?
- Für $(a, b, c) \in \mathbb{N}^3$:
Hat $a^n + b^n = c^n$ eine Lösung in den natürlichen Zahlen?
- Für ein Programm p aus der Menge aller Java-Programme:
Terminiert p ?

Beispiel 9.3 (Einige Probleme)

- Für $n \in \mathbb{N}$:
Ist n eine Primzahl?
- Für ein Wort $w \in \Sigma^*$ und
ein Element G aus der Menge aller Grammatiken über Σ :
Gilt $w \in L(G)$?
- Für ein Element G aus der Menge aller Grammatiken:
Ist $L(G)$ leer (endlich, unendlich)?
- Für $(a, b, c) \in \mathbb{N}^3$:
Hat $a^n + b^n = c^n$ eine Lösung in den natürlichen Zahlen?
- Für ein Programm p aus der Menge aller Java-Programme:
Terminiert p ?

Beispiel 9.3 (Einige Probleme)

- Für $n \in \mathbb{N}$:
Ist n eine Primzahl?
- Für ein Wort $w \in \Sigma^*$ und
ein Element G aus der Menge aller Grammatiken über Σ :
Gilt $w \in L(G)$?
- Für ein Element G aus der Menge aller Grammatiken:
Ist $L(G)$ leer (endlich, unendlich)?
- Für $(a, b, c) \in \mathbb{N}^3$:
Hat $a^n + b^n = c^n$ eine Lösung in den natürlichen Zahlen?
- Für ein Programm p aus der Menge aller Java-Programme:
Terminiert p ?

Teil II

Terminologie

- 1 Sprache, Grammatik
- 2 Warum Sprachen?
- 3 Die Chomsky-Hierarchie
- 4 Probleme über Sprachen**
- 5 Endlich, unendlich und dann?

Teil II

Terminologie

- 1 Sprache, Grammatik
- 2 Warum Sprachen?
- 3 Die Chomsky-Hierarchie
- 4 Probleme über Sprachen
- 5 Endlich, unendlich und dann?**

Definition 10.1 (Abzählbarkeit)

Eine Menge M heißt **abzählbar**, wenn

- es eine **surjektive** Funktion

$$f : \mathbb{N}_0 \rightarrow M$$

gibt,

- oder M leer ist.

Intuition

Eine Menge ist abzählbar, wenn sie höchstens so mächtig wie \mathbb{N}_0 ist.

Definition 10.1 (Abzählbarkeit)

Eine Menge M heißt **abzählbar**, wenn

- es eine **surjektive** Funktion

$$f : \mathbb{N}_0 \rightarrow M$$

gibt,

- oder M leer ist.

Intuition

Eine Menge ist abzählbar, wenn sie höchstens so mächtig wie \mathbb{N}_0 ist.

Definition 10.1 (Abzählbarkeit)

Eine Menge M heißt **abzählbar**, wenn

- es eine **surjektive** Funktion

$$f : \mathbb{N}_0 \rightarrow M$$

gibt,

- oder M leer ist.

Intuition

Eine Menge ist abzählbar, wenn sie höchstens so mächtig wie \mathbb{N}_0 ist.

Lemma 10.2

Eine Menge M ist abzählbar, wenn es eine *injektive* Funktion

$$f : M \rightarrow \mathbb{N}_0$$

gibt.

Beispiel 10.3

Abzählbar sind:

- \mathbb{N}_0
- \mathbb{Q}
- alle endlichen Mengen
- die Vereinigung zweier abzählbarer Mengen
- die Vereinigung abzählbar vieler abzählbarer Mengen

Beispiel 10.3

Abzählbar sind:

- \mathbb{N}_0
- \mathbb{Q}
- alle endlichen Mengen
- die Vereinigung zweier abzählbarer Mengen
- die Vereinigung abzählbar vieler abzählbarer Mengen

Beispiel 10.3

Abzählbar sind:

- \mathbb{N}_0
- \mathbb{Q}
- **alle endlichen Mengen**
- die Vereinigung zweier abzählbarer Mengen
- die Vereinigung abzählbar vieler abzählbarer Mengen

Beispiel 10.3

Abzählbar sind:

- \mathbb{N}_0
- \mathbb{Q}
- alle endlichen Mengen
- die Vereinigung zweier abzählbarer Mengen
- die Vereinigung abzählbar vieler abzählbarer Mengen

Beispiel 10.3

Abzählbar sind:

- \mathbb{N}_0
- \mathbb{Q}
- alle endlichen Mengen
- die Vereinigung zweier abzählbarer Mengen
- die Vereinigung abzählbar vieler abzählbarer Mengen

David Hilbert ★ 1862, † 1943

- Einer der bedeutendsten und einflußreichsten Mathematiker aller Zeiten
- Professor in Königsberg und Göttingen
- Wichtige Beiträge zu
 - Logik
 - Funktionalanalysis
 - Zahlentheorie
 - Mathematische Grundlagen der Physik
 - uvm.



David Hilbert ★ 1862, † 1943

- Einer der bedeutendsten und einflußreichsten Mathematiker aller Zeiten
- **Professor in Königsberg und Göttingen**
- Wichtige Beiträge zu
 - Logik
 - Funktionalanalysis
 - Zahlentheorie
 - Mathematische Grundlagen der Physik
 - uvm.



David Hilbert ★ 1862, † 1943

- Einer der bedeutendsten und einflußreichsten Mathematiker aller Zeiten
- Professor in Königsberg und Göttingen
- **Wichtige Beiträge zu**
 - Logik
 - Funktionalanalysis
 - Zahlentheorie
 - Mathematische Grundlagen der Physik
 - uvm.



Diagonalisierungsargument für Überabzählbarkeit

Theorem 10.4

Die Menge \mathbb{R} der reellen Zahlen ist überabzählbar.

Beweis.

Wir zeigen, dass schon das Intervall $[0, 1]$ überabzählbar ist.

Annahme: Es gibt eine Aufzählung, also eine surjektive Funktion

$$f : \mathbb{N}_0 \rightarrow [0, 1]$$

Dann sei

$$f(i) = 0, d_0^i d_1^i d_2^i \dots$$

die Dezimaldarstellung der i -ten reellen Zahl. □

Diagonalisierungsargument für Überabzählbarkeit

Theorem 10.4

Die Menge \mathbb{R} der reellen Zahlen ist überabzählbar.

Beweis.

Wir zeigen, dass schon das Intervall $[0, 1]$ überabzählbar ist.

Annahme: Es gibt eine Aufzählung, also eine surjektive Funktion

$$f : \mathbb{N}_0 \rightarrow [0, 1]$$

Dann sei

$$f(i) = 0, d_0^i d_1^i d_2^i \dots$$

die Dezimaldarstellung der i -ten reellen Zahl. □

Diagonalisierungsargument für Überabzählbarkeit

Beweis.

Fortsetzung:

Wir definieren eine neue Zahl $d = 0, \bar{d}_0 \bar{d}_1 \bar{d}_2 \dots$ durch

$$\bar{d}_n = \begin{cases} d_n^n + 1 & \text{falls } d_n^n < 9 \\ 0 & \text{sonst} \end{cases}$$

d unterscheidet sich in der n -ten Stelle von d_n .

Also $d \neq d_n$ für alle $n \in \mathbb{N}_0$

Also kommt d in der Aufzählung nicht vor. Widerspruch! □

Wieviele gibt es?

Wieviele

- Grammatiken
 - Sprachen
 - Algorithmen
- gibt es überhaupt?

Mögliche Antworten

- Endlich viele
- Unendlich viele
- Abzählbar viele
- Überabzählbar viele
- Nicht klar für Algorithmen, da dieser Begriff nicht genau definiert wurde

Wieviele gibt es?

Wieviele

- Grammatiken
 - Sprachen
 - Algorithmen
- gibt es überhaupt?

Mögliche Antworten

- **Endlich viele**
- Unendlich viele
- Abzählbar viele
- Überabzählbar viele
- Nicht klar für Algorithmen, da dieser Begriff nicht genau definiert wurde

Wieviele gibt es?

Wieviele

- Grammatiken
 - Sprachen
 - Algorithmen
- gibt es überhaupt?

Mögliche Antworten

- Endlich viele
- **Unendlich viele**
- Abzählbar viele
- Überabzählbar viele
- Nicht klar für Algorithmen, da dieser Begriff nicht genau definiert wurde

Wieviele gibt es?

Wieviele

- Grammatiken
 - Sprachen
 - Algorithmen
- gibt es überhaupt?

Mögliche Antworten

- Endlich viele
- Unendlich viele
- **Abzählbar viele**
- Überabzählbar viele
- Nicht klar für Algorithmen, da dieser Begriff nicht genau definiert wurde

Wieviele gibt es?

Wieviele

- Grammatiken
 - Sprachen
 - Algorithmen
- gibt es überhaupt?

Mögliche Antworten

- Endlich viele
- Unendlich viele
- Abzählbar viele
- **Überabzählbar viele**
- Nicht klar für Algorithmen, da dieser Begriff nicht genau definiert wurde

Wieviele gibt es?

Wieviele

- Grammatiken
 - Sprachen
 - Algorithmen
- gibt es überhaupt?

Mögliche Antworten

- Endlich viele
- Unendlich viele
- Abzählbar viele
- Überabzählbar viele
- **Nicht klar für Algorithmen, da dieser Begriff nicht genau definiert wurde**

Wieviele Wörter, Grammatiken gibt es?

Lemma 10.5

*Gegeben: Signatur Σ , endlich oder abzählbar unendlich
Dann ist Σ^* abzählbar unendlich.*

Beweis.

Σ ist abzählbar, also ist Σ^i abzählbar, $i \in \mathbb{N}$.

Σ^* ist die Vereinigung der abzählbar vielen abzählbaren Mengen Σ^i . □

Wieviele Wörter, Grammatiken gibt es?

Lemma 10.5

*Gegeben: Signatur Σ , endlich oder abzählbar unendlich
Dann ist Σ^* abzählbar unendlich.*

Beweis.

Σ ist abzählbar, also ist Σ^i abzählbar, $i \in \mathbb{N}$.

Σ^* ist die Vereinigung der abzählbar vielen abzählbaren Mengen Σ^i . □

Wieviele Wörter, Grammatiken gibt es?

Lemma 10.6

Gegeben: Signatur Σ , endlich oder abzählbar unendlich

Dann ist die Menge aller Grammatiken über Σ abzählbar unendlich

Beweis.

Grammatiken sind endlich und also als Wörter über einer geeigneten erweiterten Grammatik

$$\Sigma \cup V \cup \{\rightarrow, \dots\}$$

darstellbar.

Die Menge der Wörter über dieser erweiterten Grammatik ist abzählbar (Lemma 10.5). □

Wieviele Wörter, Grammatiken gibt es?

Lemma 10.6

Gegeben: Signatur Σ , endlich oder abzählbar unendlich

Dann ist die Menge aller Grammatiken über Σ abzählbar unendlich

Beweis.

Grammatiken sind endlich und also als Wörter über einer geeigneten erweiterten Grammatik

$$\Sigma \cup V \cup \{\rightarrow, \dots\}$$

darstellbar.

Die Menge der Wörter über dieser erweiterten Grammatik ist abzählbar (Lemma 10.5). □

Wieviele Wörter, Grammatiken gibt es?

Lemma 10.6

Gegeben: Signatur Σ , endlich oder abzählbar unendlich

Dann ist die Menge aller Grammatiken über Σ abzählbar unendlich

Beweis.

Grammatiken sind endlich und also als Wörter über einer geeigneten erweiterten Grammatik

$$\Sigma \cup V \cup \{\rightarrow, \dots\}$$

darstellbar.

Die Menge der Wörter über dieser erweiterten Grammatik ist abzählbar (Lemma 10.5). □

Wieviele Algorithmen gibt es?

Lemma 10.7

Es gibt (nur) abzählbar viele Algorithmen.

Beweis.

Algorithmen müssen **per Definition** eine endliche Beschreibung haben.

Sie sind also als Wörter über einer Signatur Σ darstellbar
(für jedes abzählbare Σ).

Also sind sie abzählbar (Lemma 10.5).

Wieviele Algorithmen gibt es?

Lemma 10.7

Es gibt (nur) abzählbar viele Algorithmen.

Beweis.

Algorithmen müssen **per Definition** eine endliche Beschreibung haben.

Sie sind also als Wörter über einer Signatur Σ darstellbar
(für jedes abzählbare Σ).

Also sind sie abzählbar (Lemma 10.5).

Wieviele Algorithmen gibt es?

Lemma 10.7

Es gibt (nur) abzählbar viele Algorithmen.

Beweis.

Algorithmen müssen **per Definition** eine endliche Beschreibung haben. Sie sind also als Wörter über einer Signatur Σ darstellbar (für jedes abzählbare Σ).

Also sind sie abzählbar (Lemma 10.5).

Wieviele Algorithmen gibt es?

Lemma 10.7

Es gibt (nur) abzählbar viele Algorithmen.

Beweis.

Algorithmen müssen **per Definition** eine endliche Beschreibung haben.

Sie sind also als Wörter über einer Signatur Σ darstellbar
(für jedes abzählbare Σ).

Also sind sie abzählbar (Lemma 10.5). □

Wieviele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ gibt es?

Lemma 10.8

Es gibt überabzählbar viele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$.

Beweis.

Angenommen, es existiere eine Abzählung

$$f_1, f_2, \dots, f_n, \dots$$

Dann sei

$$C : \mathbb{N}_0 \rightarrow \mathbb{N}_0 \quad \text{mit} \quad C(n) = \begin{cases} 1 & \text{falls } f_n(n) = 0; \\ 0 & \text{sonst} \end{cases}$$

$$C(i) \neq f_i(i)$$

Also: C ist von allen f_i verschieden.

Widerspruch!

Wieviele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ gibt es?

Lemma 10.8

Es gibt überabzählbar viele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$.

Beweis.

Angenommen, es existiere eine Abzählung

$$f_1, f_2, \dots, f_n, \dots$$

Dann sei

$$C : \mathbb{N}_0 \rightarrow \mathbb{N}_0 \quad \text{mit} \quad C(n) = \begin{cases} 1 & \text{falls } f_n(n) = 0; \\ 0 & \text{sonst} \end{cases}$$

$$C(i) \neq f_i(i)$$

Also: C ist von allen f_i verschieden.

Widerspruch!

Wieviele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ gibt es?

Lemma 10.8

Es gibt überabzählbar viele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$.

Beweis.

Angenommen, es existiere eine Abzählung

$$f_1, f_2, \dots, f_n, \dots$$

Dann sei

$$C : \mathbb{N}_0 \rightarrow \mathbb{N}_0 \quad \text{mit} \quad C(n) = \begin{cases} 1 & \text{falls } f_n(n) = 0; \\ 0 & \text{sonst} \end{cases}$$

$$C(i) \neq f_i(i)$$

Also: C ist von allen f_i verschieden.

Widerspruch!

Wieviele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ gibt es?

Lemma 10.8

Es gibt überabzählbar viele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$.

Beweis.

Angenommen, es existiere eine Abzählung

$$f_1, f_2, \dots, f_n, \dots$$

Dann sei

$$C : \mathbb{N}_0 \rightarrow \mathbb{N}_0 \quad \text{mit} \quad C(n) = \begin{cases} 1 & \text{falls } f_n(n) = 0; \\ 0 & \text{sonst} \end{cases}$$

$$C(i) \neq f_i(i)$$

Also: C ist von allen f_i verschieden.

Widerspruch!

Wieviele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ gibt es?

Lemma 10.8

Es gibt überabzählbar viele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$.

Beweis.

Angenommen, es existiere eine Abzählung

$$f_1, f_2, \dots, f_n, \dots$$

Dann sei

$$C : \mathbb{N}_0 \rightarrow \mathbb{N}_0 \quad \text{mit} \quad C(n) = \begin{cases} 1 & \text{falls } f_n(n) = 0; \\ 0 & \text{sonst} \end{cases}$$

$$C(i) \neq f_i(i)$$

Also: C ist von allen f_i verschieden.

Widerspruch!

Wieviele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ gibt es?

Lemma 10.8

Es gibt überabzählbar viele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$.

Beweis.

Angenommen, es existiere eine Abzählung

$$f_1, f_2, \dots, f_n, \dots$$

Dann sei

$$C : \mathbb{N}_0 \rightarrow \mathbb{N}_0 \quad \text{mit} \quad C(n) = \begin{cases} 1 & \text{falls } f_n(n) = 0; \\ 0 & \text{sonst} \end{cases}$$

$$C(i) \neq f_i(i)$$

Also: C ist von allen f_i verschieden.

Widerspruch!



Wieviele Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ gibt es?

Lemma 10.9

Es gibt überabzählbar viele Funktionen $f : \mathbb{N}_0 \rightarrow \{0, 1\}$.

Beweis.

Analog. □

Wieviele Sprachen gibt es?

Lemma 10.10

Gegeben eine Signatur Σ (endlich oder unendlich).

Die Menge der Sprachen über Σ ist überabzählbar.

Beweis.

Sei eine beliebige Abzählung aller Wörter über Σ gegeben:

$$w_1, w_2, \dots$$

Dann kann man die Sprachen L über Σ mit den Funktionen $f : \mathbb{N}_0 \rightarrow \{0, 1\}$ identifizieren, vermittelt

$$f(i) = 1 \quad \text{gdw} \quad w_i \in L$$

Von diesen gibt es überabzählbar viele. □

Wieviele Sprachen gibt es?

Lemma 10.10

Gegeben eine Signatur Σ (endlich oder unendlich).

Die Menge der Sprachen über Σ ist überabzählbar.

Beweis.

Sei eine beliebige Abzählung aller Wörter über Σ gegeben:

$$w_1, w_2, \dots$$

Dann kann man die Sprachen L über Σ mit den Funktionen $f : \mathbb{N}_0 \rightarrow \{0, 1\}$ identifizieren, vermittels

$$f(i) = 1 \quad \text{gdw} \quad w_i \in L$$

Von diesen gibt es überabzählbar viele. □

Wieviele Sprachen gibt es?

Lemma 10.10

Gegeben eine Signatur Σ (endlich oder unendlich).

Die Menge der Sprachen über Σ ist überabzählbar.

Beweis.

Sei eine beliebige Abzählung aller Wörter über Σ gegeben:

$$w_1, w_2, \dots$$

Dann kann man die Sprachen L über Σ mit den Funktionen $f : \mathbb{N}_0 \rightarrow \{0, 1\}$ identifizieren, vermittels

$$f(i) = 1 \quad \text{gdw} \quad w_i \in L$$

Von diesen gibt es überabzählbar viele. □

Wieviele Sprachen gibt es?

Lemma 10.10

Gegeben eine Signatur Σ (endlich oder unendlich).

Die Menge der Sprachen über Σ ist überabzählbar.

Beweis.

Sei eine beliebige Abzählung aller Wörter über Σ gegeben:

$$w_1, w_2, \dots$$

Dann kann man die Sprachen L über Σ mit den Funktionen $f : \mathbb{N}_0 \rightarrow \{0, 1\}$ identifizieren, vermittels

$$f(i) = 1 \quad \text{gdw} \quad w_i \in L$$

Von diesen gibt es überabzählbar viele. □

Korollar 10.11

Nicht jede Sprache kann durch eine Grammatik dargestellt werden.

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- \mathbb{N}
- Menge aller Wörter
- Menge aller Grammatiken
- Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von \mathbb{N}
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ bzw. $f: \mathbb{N}_0 \rightarrow \{0, 1\}$
- Die Menge aller Sprachen

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- \mathbb{N}
 - Menge aller Wörter
 - Menge aller Grammatiken
 - Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von \mathbb{N}
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ bzw. $f: \mathbb{N}_0 \rightarrow \{0, 1\}$
- Die Menge aller Sprachen

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- \mathbb{N}
- Menge aller Wörter
- Menge aller Grammatiken
- Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von \mathbb{N}
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ bzw. $f: \mathbb{N}_0 \rightarrow \{0, 1\}$
- Die Menge aller Sprachen

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- \mathbb{N}
- Menge aller Wörter
- Menge aller Grammatiken
- Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von \mathbb{N}
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ bzw. $f: \mathbb{N}_0 \rightarrow \{0, 1\}$
- Die Menge aller Sprachen

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- \mathbb{N}
- Menge aller Wörter
- Menge aller Grammatiken
- Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von \mathbb{N}
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ bzw. $f: \mathbb{N}_0 \rightarrow \{0, 1\}$
- Die Menge aller Sprachen

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- \mathbb{N}
- Menge aller Wörter
- Menge aller Grammatiken
- Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von \mathbb{N}
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ bzw. $f: \mathbb{N}_0 \rightarrow \{0, 1\}$
- Die Menge aller Sprachen

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- \mathbb{N}
- Menge aller Wörter
- Menge aller Grammatiken
- Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von \mathbb{N}
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ bzw. $f: \mathbb{N}_0 \rightarrow \{0, 1\}$
- Die Menge aller Sprachen

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- \mathbb{N}
- Menge aller Wörter
- Menge aller Grammatiken
- Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von \mathbb{N}
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ bzw. $f: \mathbb{N}_0 \rightarrow \{0, 1\}$
- Die Menge aller Sprachen

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- \mathbb{N}
- Menge aller Wörter
- Menge aller Grammatiken
- Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von \mathbb{N}
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ bzw. $f: \mathbb{N}_0 \rightarrow \{0, 1\}$
- Die Menge aller Sprachen

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- \mathbb{N}
- Menge aller Wörter
- Menge aller Grammatiken
- Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von \mathbb{N}
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ bzw. $f : \mathbb{N}_0 \rightarrow \{0, 1\}$
- Die Menge aller Sprachen

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- \mathbb{N}
- Menge aller Wörter
- Menge aller Grammatiken
- Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von \mathbb{N}
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ bzw. $f : \mathbb{N}_0 \rightarrow \{0, 1\}$
- **Die Menge aller Sprachen**