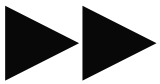
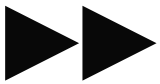
Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Gliederung der Vorlesung

0. Grundbegriffe
1. Formale Sprachen/Automatentheorie
    - **1.1. Grammatiken**
    - 1.2. Reguläre Sprachen
    - 1.3. Kontext-freie Sprachen
2. Berechnungstheorie
    - 2.1. Entscheidungsprobleme
    - 2.2. Berechenbarkeitsmodelle
    - 2.3. Die Churchsche These
    - 2.4. Unentscheidbarkeit
3. Komplexitätstheorie
    - 3.1. Nicht-deterministische Turing Maschinen
    - 3.1 Komplexitätsmaße
    - 3.2. Das P=NP? Problem

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Grammatiken

Zielstellung

Menge der zulässigen Ausdrücke

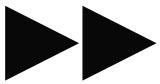
- einer Programmiersprache
- einer Seitenbeschreibungssprache
- eines Datenaustauschformats
- ...

charakterisieren

Randbedingungen

- verständlich (/* für den Menschen */)
- verarbeitbar (/* von der Maschine */)

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Grammatiken

Beispiel: Bezeichner in der Programmiersprache C++

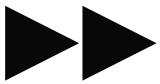
Vom Programmierer definierte Größen werden durch Namen angesprochen.
Diese Namen werden nach folgenden Regeln gebildet:

- Sie dürfen mit einem Buchstaben oder mit einem _ beginnen.
- Sie dürfen beliebige Zeichen (/* außer dem Leerzeichen */) enthalten.
- Sie dürfen keine Schlüsselwörter überdecken.

... es wäre zu klären, was ist/sind

- Buchstaben/Zeichen
- Menge der zulässigen Schlüsselwörter
- Namen, die Schlüsselwörter überdecken

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Grammatiken

Beispiel: Bezeichner in der Programmiersprache C++

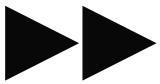
Verständlichkeit
- alles klar

Verarbeitbarkeit
- ???

man benötigt ein Programm, welches
- Zeichenketten verarbeiten kann und „herausfindet“, ob eine gegebene Zeichenkette gemäß der präzisierten Regeln gebildet wurde

... eigentlich sollte solch ein Programm automatisch aus den Regeln generiert werden können

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Grammatiken

Beispiel: Arithmetische Ausdrücke

Arithmetische Ausdrücke werden nach den folgenden Regeln gebildet:

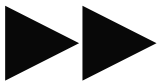
- Jede Variablenname ist ein arithmetischer Ausdruck.
- Wenn A1 und A2 arithmetische Ausdrücke sind, so sind auch (A1 + A2) und (A1 * A2) arithmetische Ausdrücke.

... es wäre zu klären, was sind

- Variablennamen

... Anmerkungen wie zuvor

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Grammatiken

... ein wenig abstrakter

es wäre zu präzisieren

- welche Zeichen überhaupt vorkommen dürfen
- welche Zeichenketten als zulässige Ausdrücke in Frage kommen

- gegeben ein Alphabet Σ
- Menge Σ^* aller Zeichenketten über dem Alphabet
- Menge L aller zulässigen Zeichenketten, d.h. eine bestimmte Teilmenge von Σ^*

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

... ein wenig abstrakter

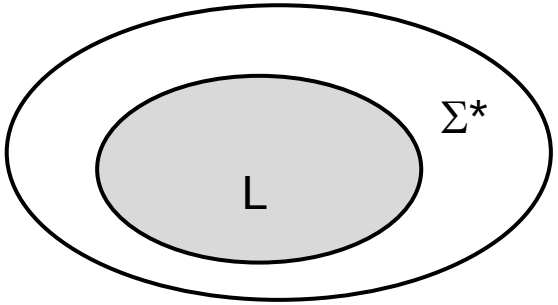
zentrale konzeptionelle Frage

- wie charakterisiert man die interessierenden Teilmengen L von Σ^*

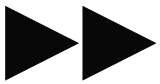
zentrale algorithmische Frage

- wie sieht ein Programm aus, welches – gegeben irgendeine Zeichenkette w aus Σ^* – entscheidet, ob w zu L gehört oder nicht
- gibt es so ein Programm eigentlich immer

(/* Begriff: Wort bzw. Membership-Problem */)



Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



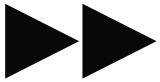
Grammatiken

... konzeptionelle Frage

Bestimmungsstücke einer Chomsky-Grammatik

- ein endliches Alphabet Σ von Symbolen (/* Terminalsymbole */)
- ein endliches Alphabet V von Variablen (/* Hilfssymbole */)
- ein ausgezeichnetes Startsymbol $S \in V$
- eine endliche Menge von Regeln (/* Produktionen */); d.h. Paare (l,r) mit $l \in (\Sigma \cup V)^* \setminus \{ \varepsilon \}$ und $r \in (\Sigma \cup V)^*$

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Grammatiken

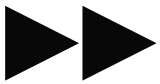
Beispiel: Bezeichner in der Programmiersprache C++

Σ = { a, 1, _ }
V = { S, H }
S
R = { (S,a), (S,_H), (S,aH),
(H,aH),(H,1H), (H,a), (H,1) }

... andere Notation

S → a
S → _H
S → aH
H → aH
H → 1H
H → a
H → 1

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Grammatiken

Beispiel: Bezeichner in der Programmiersprache C++

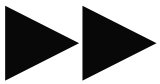
Σ = { a, 1, _ }
V = { S, H }
S

S → a
S → _H
S → aH
H → aH
H → 1H
H → a
H → 1

beschriebene Sprache

L(G) = { a,
_a, _1, aa, a1,
_aa, _a1, _1a, _11, aaa, ..., a11,
... }

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Grammatiken

Beispiel: arithmetische Ausdrücke

Σ = { a, + ,*, (,) }
V = { S }
S

S → a
S → (S+ S)
S → (S*S)

beschriebene Sprache

L(G) = { a,
(a+a), (a*a),
(a+(a+a)),(a+(a*a)), ..., ((a*a)*a),
... }

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

die von einer Grammatik definierte Ableitungsrelation $\rightarrow_G$

- es sei G = [Σ,V,S,R] eine Grammatik
- es seien u,v $\in (\Sigma \cup V)^*$
- u $\rightarrow_G$ v, falls es Zeichenketten x,y $\in (\Sigma \cup V)^*$ und eine Regel (l,r) $\in$ R, gibt, so daß gilt:
  - u = xly und v = xry

Σ = { 0, 1 }
V = { S }
S

S → 0S
S → 1S
S → 111

00S0SS $\rightarrow_G$ 00S0111S
S $\rightarrow_G$ 111

~~111 $\rightarrow_G$ 1111~~
~~1S0S $\rightarrow_G$ 111010~~

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

die von einer Grammatik definierte Sprache L(G)

- es sei G = [Σ,V,S,R] eine Grammatik
- es sei u $\in$ ($\Sigma \cup$ V)*
- u $\in$ L(G), falls u $\in \Sigma^*$ und S $\rightarrow^*_G$ u

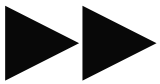
---

Σ = { 0, 1 }
V = { S }
S

S → 0S
S → 1S
S → 111

S $\rightarrow^*_G$ 111
S $\rightarrow^*_G$ 00111

~~S $\rightarrow^*_G$ 11~~
~~S $\rightarrow^*_G$ 111010~~



Grammatiken

Beispiel

L = { v | v ist Binärdarstellung einer durch 4 teilbaren Zahl }

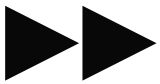
Beobachtungen

- v besteht nur aus den Buchstaben 0 bzw. 1
- v beginnt mit einer 1
- die letzten beiden Ziffern von v sind 0‘en

Σ = { 0, 1 }
V = { S, S‘ }
S

S → 1S‘
S‘ → 0S‘
S‘ → 1S‘
S‘ → 00

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Grammatiken

Beispiel

$L = \{ 0^n 1^n \mid n \geq 1 \}$

... korrekt geklammerte Ausdrücke

$\Sigma = \{ 0, 1 \}$
$V = \{ S \}$
S

$S \rightarrow 01$
$S \rightarrow 0S1$

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

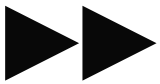
Grammatiken

Beispiel

$L = \{ 0^n 1^n 0^n \mid n \geq 1 \}$

$\Sigma = \{ 0, 1 \}$
$V = \{ S, S', H \}$
S

$S \rightarrow 010$
$S \rightarrow 0S'10$
$S' \rightarrow 01H$
$S' \rightarrow 0S'1H$
$H1 \rightarrow 1H$
$H0 \rightarrow 00$



Grammatiken

... algorithmische Frage

es sei G = [Σ,V,S,R] eine Grammatik

... offenbar kann man – unter Verwendung der Regeln der Grammatik – ganz einfach ein Programm angeben, welches folgendes leistet:

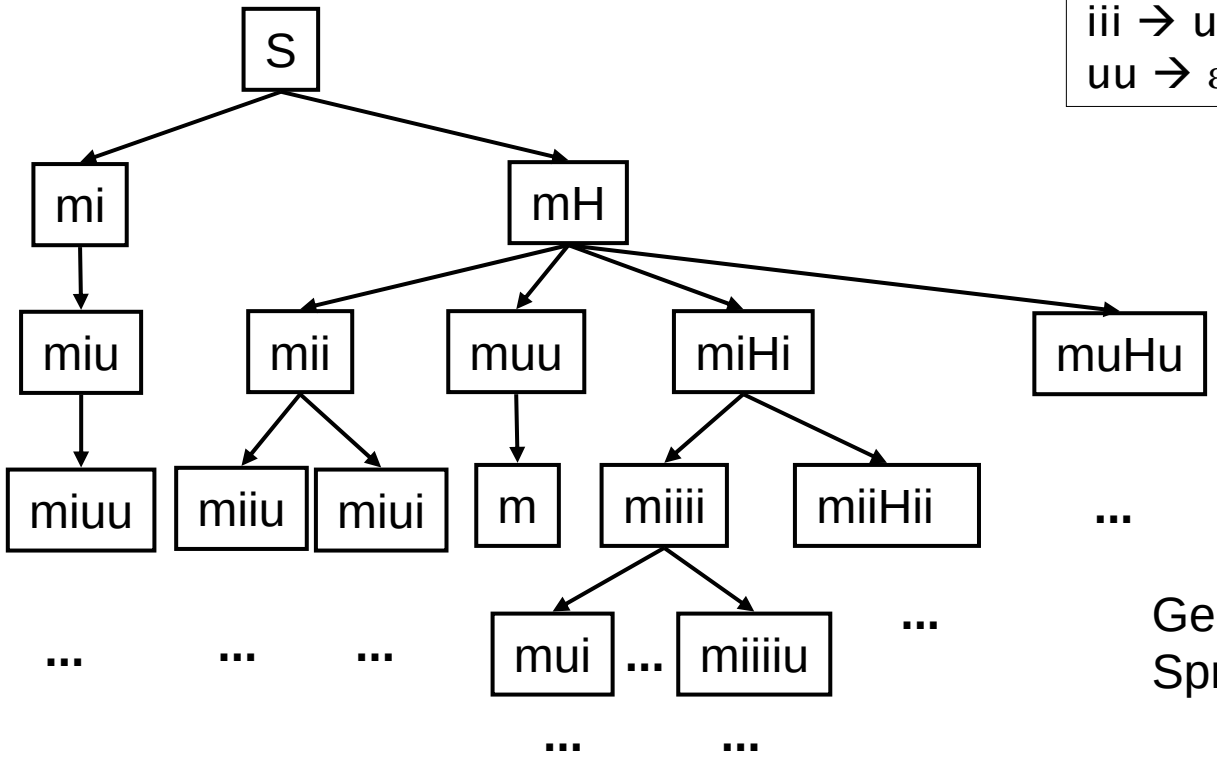
- das Programm bestimmt alle Wörter der Sprache L(G) und schreibt diese in eine Datei

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

Σ = { m, i, u }
V = { S, H }
S

S → mi	H → ii
S → mH	H → uu
	H → iHi
i→ iu	H → uHu
iii → u	
uu → ε	



Gehört das Wort mu zur Sprache L(G)?

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

Σ = { m, i, u }
V = { S, H }
S

S → mi	H → ii
S → mH	H → uu
	H → iHi
i→ iu	H → uHu
iii → u	
uu → ε	

- durch sukzessives Ableiten kann man alle Wörter, die zur Sprache L(G) gehören, ableiten
- um herauszubekommen, ob ein Wort nicht zur Sprache L(G) gehört, muß man wissen, wann man mit dem Ableiten aufhören kann

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

$\Sigma = \{ m, i, u \}$
$V = \{ S, H \}$
S

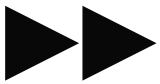
$S \rightarrow mi$	$H \rightarrow ii$
$S \rightarrow mH$	$H \rightarrow uu$
	$H \rightarrow iHi$
$i \rightarrow iu$	$H \rightarrow uHu$
$iii \rightarrow u$	
$uu \rightarrow \varepsilon$	

Gehört das Wort mu zur Sprache L(G)?

ja, denn

$$S \rightarrow_G mH \rightarrow_G miHi \rightarrow_G miiHii$$
$$\rightarrow_G miiiiii \rightarrow_G miiiiiiu$$
$$\rightarrow_G muiiiu \rightarrow_G muuu \rightarrow_G mu$$

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



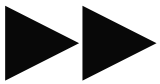
Grammatiken

... algorithmische Frage

es sei G = [Σ,V,S,R] eine Grammatik

G heißt kontext-sensitive Grammatik, falls für alle Regeln (l,r) $\in$ R gilt:

- $|l| \leq |r|$



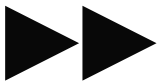
Grammatiken

... Vorteil der Einschränkung auf kontext-sensitive Grammatiken

Es gibt einen Algorithmus, der – bei Eingabe einer kontext-sensitiven Grammatik $G = [\Sigma,V,S,R]$ und eines Wortes $w \in \Sigma^*$ – eine „1“ ausgibt, falls $w \in L(G)$ gilt, und eine „0“ ausgibt, falls $w \notin L(G)$ gilt.

... dieser Algorithmus entscheidet das Membership-Problem für die entsprechende Sprache L(G)

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Grammatiken

... algorithmische Idee, um w $\in$ L(G) zu entscheiden

es sei G = [Σ,V,S,R] eine kontext-sensitive Grammatik
es sei w ein Wort aus mit |w| = n

Schritt 0:
- setze Y = { S } und führe Schritt 1 aus

Schritt m (m $\geq$ 1):
- bestimme die Menge Y‘ aller Zeichenketten, die sich aus den Zeichenkette in Y in einem Schritt ableiten lassen und eine Länge kleiner gleich n haben
- falls w $\in$ Y‘, so gib „1“ aus
- falls w $\notin$ Y‘ und Y‘ = Y, so gib „0“ aus
- sonst setze Y = Y‘ und führe Schritt m+1 aus

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

... Illustration (/* Schritt 0 */)

Σ = { 0, 1 }
V = { S }
S

S → 01
S → 0S1

S

w = 010100

Anmerkung: zu Y gehören die Wörter in den weiß hinterlegten Feldern

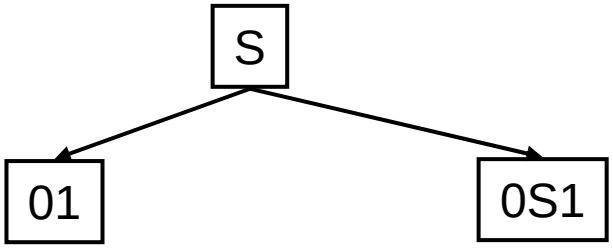
Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

... Illustration (/* Schritt 1 */)

Σ = { 0, 1 }
V = { S }
S

S → 01
S → 0S1



w = 010100

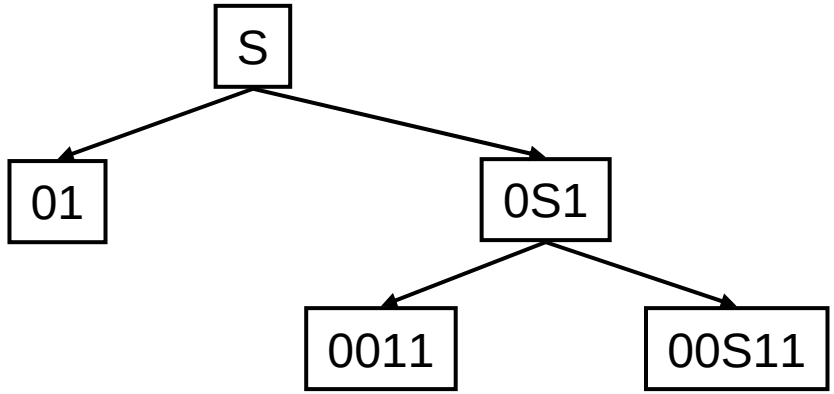
Anmerkung: zu Y gehören die Wörter in den weiß hinterlegten Feldern

Grammatiken

... Illustration (/* Schritt 2 */)

Σ = { 0, 1 }
V = { S }
S

S → 01
S → 0S1



w = 010100

Anmerkung: zu Y gehören die Wörter in den weiß hinterlegten Feldern

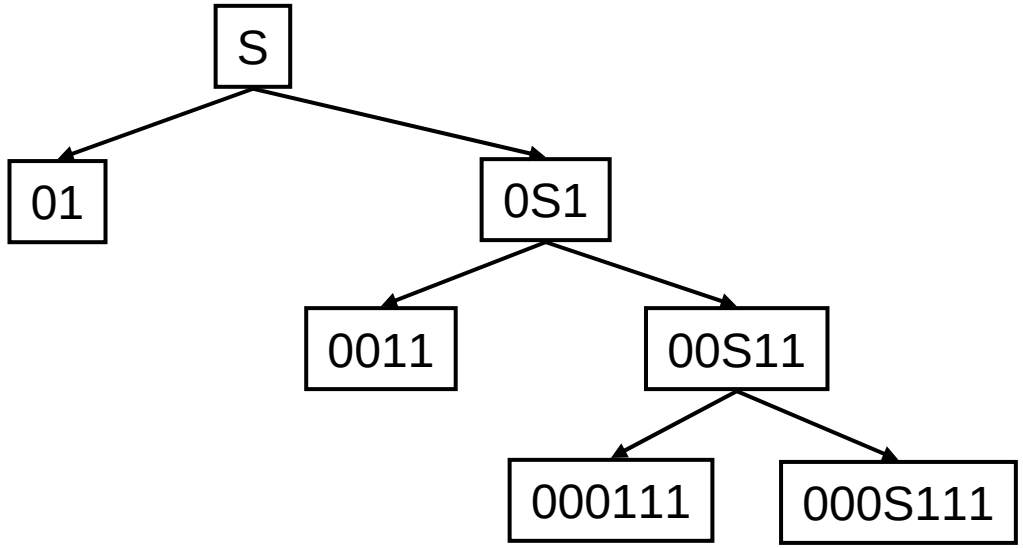
Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

... Illustration (/* Schritt 3 */)

Σ = { 0, 1 }
V = { S }
S

S → 01
S → 0S1



w = 010100

Anmerkung: zu Y gehören die Wörter in den weiß hinterlegten Feldern

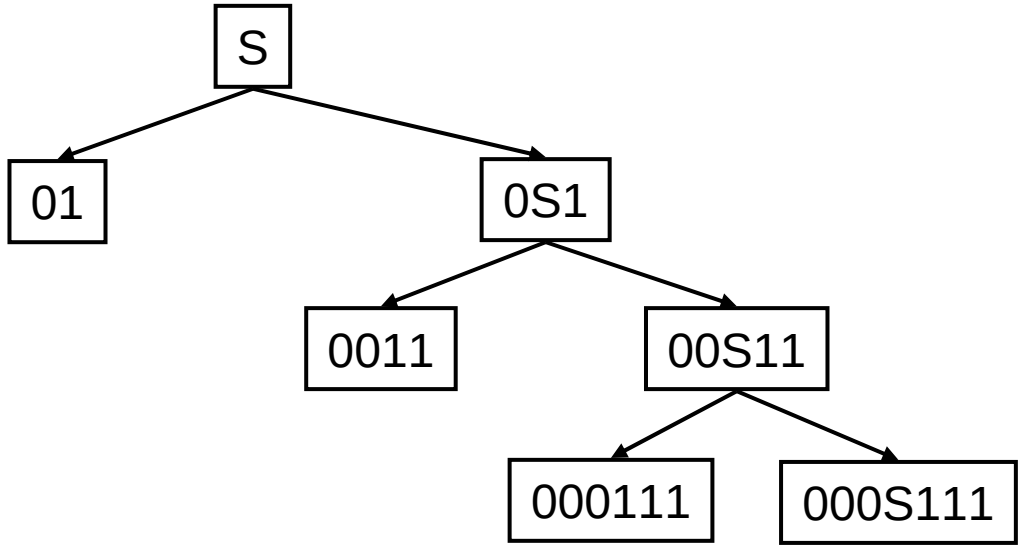
Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

... Illustration (/* Schritt 4 */)

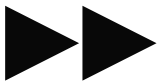
Σ = { 0, 1 }
V = { S }
S

S → 01
S → 0S1



w = 010100

Anmerkung: zu Y gehören die Wörter in den weiß hinterlegten Feldern



Grammatiken

... Beurteilung der algorithmischen Idee

- nach endlich vielen Schritten wird ein korrektes Ergebnis ausgegeben
  (/* Grund: es gibt überhaupt nur endlich viele Zeichenketten der Länge kleiner gleich |w|; in jedem Schritt muß wenigstens eine neue Zeichenkette zur Menge Y hinzukommen; es werden sukzessive alle ableitbaren Zeichenketten der Länge |w| zu Y hinzugefügt */)
- aber, der Algorithmus ist hoffnungslos ineffizient

... es ist kein qualitativ besserer Algorithmus bekannt und man geht davon aus, daß es keinen besseren gibt

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

... sinnvolle Einschränkungen der zulässigen Grammatiken

es sei G = [Σ,V,S,R] eine Grammatik

G heißt kontext-freie Grammatik, falls für alle Regeln (l,r) $\in$ R gilt:

- $|l| \leq |r|$
- l = x mit x $\in$ V

G heißt reguläre Grammatik, falls für alle Regeln (l,r) $\in$ R gilt:

- $|l| \leq |r|$
- l = x mit x $\in$ V
- r = a oder r = ax mit a $\in \Sigma$ und x $\in$ V

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Grammatiken

- Beispiel für eine kontext-sensitive Grammatik

S → 010
S → 0S‘10
S‘ → 01H
S‘ → 0S‘1H
H1 → 1H
H0 → 00

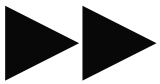
- Beispiel für eine kontext-freie Grammatik

S → 01
S → 0S1

- Beispiel für eine reguläre Grammatik

S → 0S
S → 1S
S → 1S‘
S‘ → 1S‘‘
S‘‘ → 1

~~S → 0S~~
~~S → 1S~~
~~S → 111~~



Grammatiken

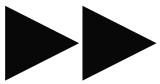
Problem am Rande

Regeln (l,r) mit $|l| \leq |r|$ taugen nicht zum Ableiten des leeren Wortes ...

Idee

bilde erst eine Grammatik, die alle nicht-leeren Wörter erzeugt, und modifiziere die Grammatik wie folgt

- wähle eine Variable H mit $H \notin V$
- ersetze in allen Regeln der Grammatik das Startsymbol S durch die Variable H
- füge die Regeln (S,ε) und (S,H) zur Grammatik hinzu



Grammatiken

Zwischenfazit

- Das Membership-Problem für jede mit einer kontext-sensitiven (kontext-freien bzw. regulären) Grammatik beschreibbaren Sprache ist entscheidbar.
- Es gibt Chomsky-Grammatiken G, so daß das Membership-Problem für die Sprache L(G) nicht entscheidbar ist.
- Der vorgestellte Algorithmus hat offenbar exponentielle Laufzeit.

... für kontext-freie und reguläre Sprachen gibt es „effizientere“ Algorithmen, mit denen das Membership-Problem entschieden werden kann