Informatik III - Tutorium XIII & XV (SR -107)

Tut Nr. 10 – Üb9, Chomsky-Hierarchie

David Münch

Universität Karlsruhe (TH)
Institut für Informatik
ITI Wagner

16. Januar 2008

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Organisatorisches

Email: muenchdavid@gmail.com

https://www.stud.uni-karlsruhe.de/~uhbro/

Tutorium 13: Mittwochs 8:00 Uhr - Raum -107
Tutorium 15: Mittwochs 9:45 Uhr - Raum -107

Übungsblattabgabe Donnerstag.

Was wollen wir heute erreichen?

Was wollen wir heute erreichen?

- Überblick über die Chomksy-Hierarchie erhalten

Was wollen wir heute erreichen?

- Überblick über die Chomksy-Hierarchie erhalten
- Umformen von Typ-3-Grammatike $\Leftrightarrow$ endlicher Automat

Was wollen wir heute erreichen?

- Überblick über die Chomksy-Hierarchie erhalten
- Umformen von Typ-3-Grammatike $\Leftrightarrow$ endlicher Automat
- Beweisen, dass eine Grammatik eine konkrete Sprache erzeugt.

Aufgabe 5

Zeige, dass es keinen polynomiellen Approximationsalgorithmus mit absoluter Gütegarantie für $\mathrm{MAX2SAT}$ gibt, falls $\mathcal{P} \neq \mathcal{NP}$.

Aufgabe 5

Das Optimalwertproblem von $\mathrm{MAX2SAT}$ ist gegeben durch eine Variablenmenge V und eine $2\mathrm{SAT}$-Klauselmenge C über V. Für eine Instanz I ist die maximale Anzahl an Klauseln $OPT(I)$ gesucht, die gleichzeitig erfüllbar sind.
Beim Entscheidungsproblem von $\mathrm{MAX2SAT}$ ist zusätzlich noch eine natürliche Zahl K gegeben. Es soll entschieden werden, ob $OPT(I) \geq K$. Es ist aus der Vorlesung bekannt, dass $\mathrm{MAX2SAT}$ $\mathcal{NP}$-vollständig ist.
Die Beweisidee:

Aufgabe 5

Das Optimalwertproblem von $\mathrm{MAX2SAT}$ ist gegeben durch eine Variablenmenge V und eine $2\mathrm{SAT}$-Klauselmenge C über V. Für eine Instanz I ist die maximale Anzahl an Klauseln $OPT(I)$ gesucht, die gleichzeitig erfüllbar sind.
Beim Entscheidungsproblem von $\mathrm{MAX2SAT}$ ist zusätzlich noch eine natürliche Zahl K gegeben. Es soll entschieden werden, ob $OPT(I) \geq K$. Es ist aus der Vorlesung bekannt, dass $\mathrm{MAX2SAT}$ $\mathcal{NP}$-vollständig ist.
Die Beweisidee: Man zeigt, dass ein polynomialer Approximationsalgorithmus zu einem optimalen, exakten, polynomialen Algorithmus abgeleitet werden kann, was ein Widerspruch zur Annahme $\mathcal{P} \neq \mathcal{NP}$ ergibt.

Aufgabe 5

Das Optimalwertproblem von $\mathrm{MAX2SAT}$ ist gegeben durch eine Variablenmenge V und eine $2\mathrm{SAT}$-Klauselmenge C über V. Für eine Instanz I ist die maximale Anzahl an Klauseln $OPT(I)$ gesucht, die gleichzeitig erfüllbar sind.
Beim Entscheidungsproblem von $\mathrm{MAX2SAT}$ ist zusätzlich noch eine natürliche Zahl K gegeben. Es soll entschieden werden, ob $OPT(I) \geq K$. Es ist aus der Vorlesung bekannt, dass $\mathrm{MAX2SAT}$ $\mathcal{NP}$-vollständig ist.
Die Beweisidee: Man zeigt, dass ein polynomialer Approximationsalgorithmus zu einem optimalen, exakten, polynomialen Algorithmus abgeleitet werden kann, was ein Widerspruch zur Annahme $\mathcal{P} \neq \mathcal{NP}$ ergibt.
Wir nehmen an es gibt einen absoluten Approximationsalgorithmus A für das Optimalwertproblem, für diesen gilt:

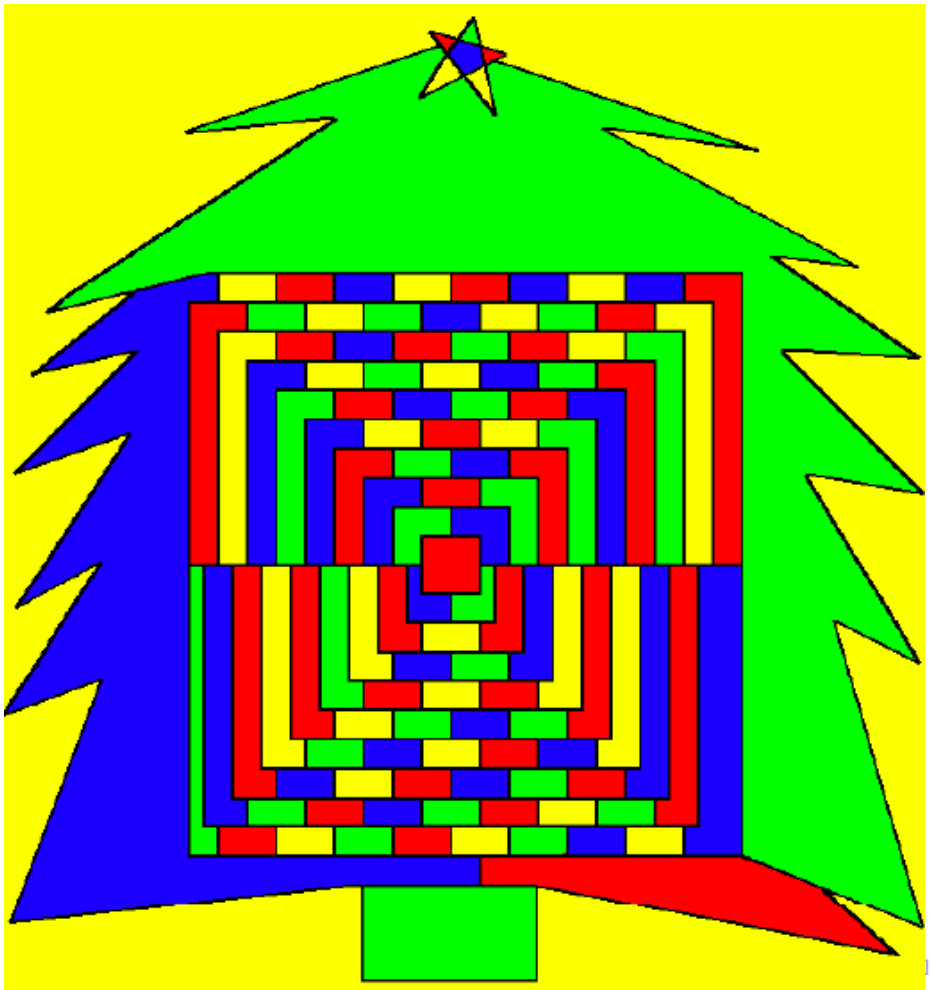
Aufgabe 5

$|A(I) - OPT(I)| \leq m$ mit $m \in \mathbb{N}$ und alle Instanzen I.
Für eine gegebene Instanz $I = (V, C, K)$ des Entscheidungsproblems konstruieren wir eine Instanz $I' = (V', C')$ des Optimalwertproblems, wobei V' $2m + 1$ mal so viele Variablen enthält wie V und sich C' ergibt, indem die Klauseln aus C $(2m + 1)$ mal durch Substitution geklont werden. Diese Transformation ist offensichtlich polynomial.
Es gilt dann:
$|A(I') - OPT(I')| \leq m$
$|A(I') - (2m + 1)OPT(I)| \leq m$
$|\frac{A(I')}{2m+1} - OPT(I)| \leq \frac{m}{2m+1} < \frac{1}{2}$
D.h. man kann $OPT(I)$ berechnen, indem man $A(I')/(2m + 1)$ rundet. Damit kann das Entscheidungsproblem in polynomieller Zeit gelöst werden.

Aufgabe 6

Ein *planarer Graph* ist ein Graph, der so in der Ebene gezeichnet werden kann, dass sich keine zwei Kanten kreuzen. Die *Facetten* eines planaren Graphen bezüglich einer gegebenen Einbettung sind die ‘maximalen, durch Kanten abgeschlossenen Flächen’.
Insbesondere wird das ‘den Graphen umgebende Gebiet’ als *äussere Facette* bezeichnet. Zwei Facetten sind *adjazent*, falls sie durch eine gemeinsame Kante begrenzt werden.
Die nachfolgende Zeichnung ist als planarer Graph zu betrachten, wobei die Knoten implizit als Schnitt- bzw Berührpunkte der Kanten gegeben seien.
Färbe die Facetten des Graphen mit vier Farben so, dass keine zwei adjazenten Facetten dieselbe Farbe haben.

Aufgabe 6



Aufgabe 7

Das Ziel der Suchproblemformulierung von SAT ist es, eine Wahrheitsbelegung zu finden, die möglichst viele Klauseln gleichzeitig erfüllt.
Algorithmus 1 löst das Problem approximativ mit relativer Gütegarantie. Gib eine (möglichst gute) Gütegarantie für den Algorithmus an und beweise diese.

Algorithm 1: GREEDY SAT

input : Klauselmenge C über den boolschen Variablen V
output: Wahrheitsbelegung $f : V \rightarrow \{true, false\}$
repeat
  l := ein beliebiges Literal, dass in maximal vielen Klauseln in C vorkommt
  C_l := die Menge der Klauseln aus C in der l vorkommt
  $C_{\overline{l}}$:= die Menge der Klauseln aus C in der $\overline{l}$ vorkommt
  v_l := die zu l gehörige Variable
  if l *ist positiv* **then**
    $f(v) := true$
    $C := C \setminus C_l$
    Lösche $\overline{l}$ aus allen Klauseln in $C_{\overline{l}}$
    Lösche alle leeren Klauseln aus C
  else
    $f(v_l) := false$
    $C := C \setminus C_{\overline{l}}$
    Lösche l aus allen Klauseln in C_l
    Lösche alle leeren Klauseln aus C
until $C = \emptyset$
OUTPUT f

Aufgabe 7

Der Algorithmus ist 2-approximativ. Wir beweisen durch Induktion nach der Anzahl der Variablen, dass der Algorithmus GREEDY SAT immer mindestens $|C/2|$ der Klauseln erfüllt. Weil es nur $|C|$ Klauseln gibt, ist dies eine obere Schranke für den Wert einer optimalen Lösung. Daraus folgt die Behauptung.

Aufgabe 7

Der Algorithmus ist 2-approximativ. Wir beweisen durch Induktion nach der Anzahl der Variablen, dass der Algorithmus GREEDY SAT immer mindestens $|C/2|$ der Klauseln erfüllt. Weil es nur $|C|$ Klauseln gibt, ist dies eine obere Schranke für den Wert einer optimalen Lösung. Daraus folgt die Behauptung.
Beweis:
Induktionsanfang: $n = 1$: Für den Fall, dass es nur eine Variable gibt, folgt die Behauptung trivial.
Induktionsvoraussetzung: $n - 1$: Es wird angenommen, die Behauptung ist für $n - 1$ Variablen bewiesen.

Aufgabe 7

Induktionsschritt: n: Von $\mathrm{GREEDY\ SAT}$ werde als erstes die Variable l betrachtet. Es seien $|C_l|$ und $|C_{\bar{l}}|$ die Anzahl der Klauseln in denen l und $\bar{l}$ vorkommen. ObdA betrachten wir $|C_l| \geq |C_{\bar{l}}|$, also setzt $\mathrm{GREEDY\ SAT}$ l auf true. Nach dieser Zuweisung müssen noch $L \geq |C| - |C_l| - |C_{\bar{l}}|$ viele Klauseln (mit $n-1$ Variablen) betrachtet werden.
Nach der Induktionsvoraussetzung werden daher wenigstens $\frac{L}{2} \geq \frac{|C|-|C_l|-|C_{\bar{l}}|}{2}$ von den L vielen Klauseln erfüllt.
Insgesamt werden daher wenigstens $|C_l| + \frac{|C|-|C_l|-|C_{\bar{l}}|}{2} = \frac{|C|+|C_l|-|C_{\bar{l}}|}{2} \geq |\frac{C}{2}|$ Klauseln der ursprünglichen Klauselmenge erfüllt. $\square$

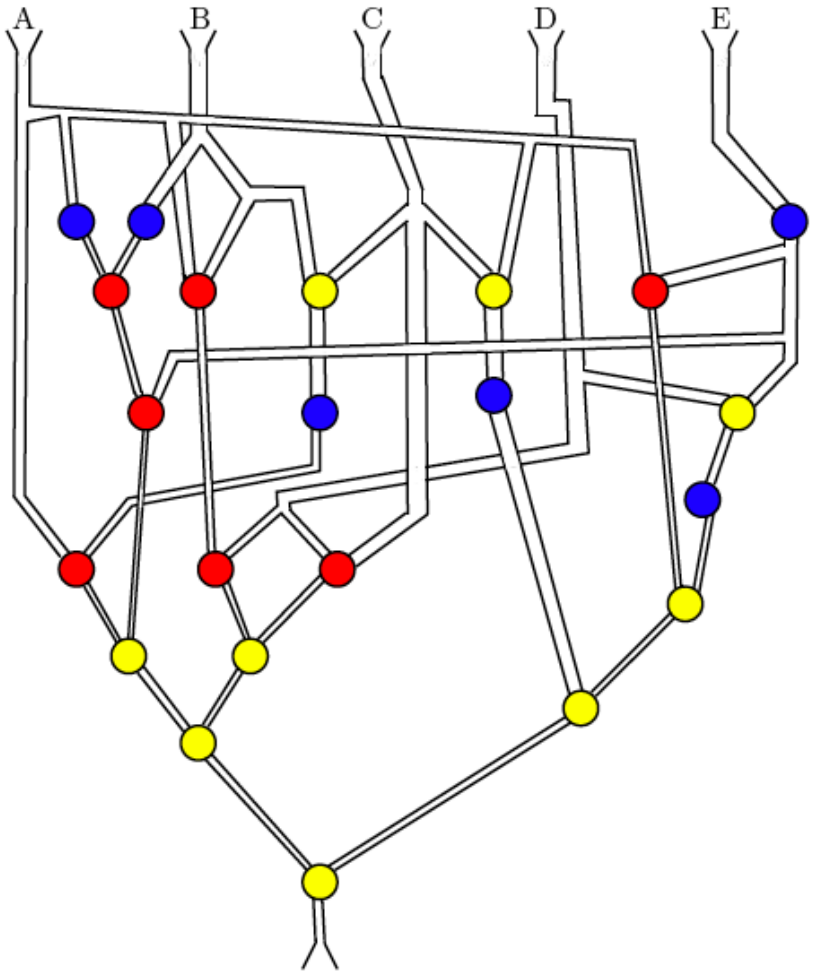
Aufgabe 8

Zu Weihnachten bekommt das Christkind Lust auf seine bekannt köstliche Schokolade. Leider hat es davon nur noch ein paar Krümel übrig, aber es hat ja seine seltsame Schokoladenmaschine, und mit ein bisschen Glück liefert diese ihm jede Menge Schokolade:
In die Trichter A bis E muss man entweder Schlangengift oder Schokolade schütten, aber nur zu oft liefert die Maschine leider bloss Schlangengift.
Nur bei einer bestimmten Verteilung von Schlangengift und Schokolade in die Trichter liefert die Maschine die begehrte Schokolade. Dass sie meistens nur Schlangengift ausspuckt, liegt daran, dass in ihr an bestimmten Stellen Kobolde ihr Unwesen treiben. Von diesen gibt es drei Sorten:

Aufgabe 8

Die **blauen Kobolde** sind von schlichtem Gemüt: Fliesst solch einem Kobold von oben Schlangengift zu, so macht er daraus Schokolade und gibt sie nach unten weiter. Entsprechend verwandelt er Schokolade in Schlangengift.
Die **roten Kobolde** arbeiten schon raffinierter: Jeder rote Kobold fängt in jeder Hand Schlangengift oder Schokolade auf. Aber nur, wenn er gleichzeitig in beiden Händen Schlangengift auffängt, gibt er auch Schlangengift ab, in allen anderen denkbaren Fällen produziert er Schokolade.
Die **gelben Kobolde** sind da anspruchsvoller: Dann und nur dann, wenn ein gelber Kobold in jeder seiner Hände von oben Schokolade auffängt, gibt er Schokolade ab, sonst nur Schlangengift.
Wo sich welche Kobolde in der Maschine eingenistet haben, kann man der folgenden Zeichnung entnehmen:

Augabe 8



Aufgabe 8

Welcher Trichter ist mit Schlangengift bzw Schokoladenkrümeln zu füttern, damit die Maschine Schokolade liefert? Stelle dazu erst eine aussagenlogische Formel auf und vereinfache diese. Denke an deinen Tutor und schreibe auch genügend Zwischenschritte auf. Hinweis: Ausprobieren bestimmter Belegungen ist nicht erforderlich.

Aufgabe 8

Die Maschine lässt sich als logische Schaltung interpretieren. Dann entsprechen:
Schokolade $\Leftrightarrow$ logische 1
Schlangengift $\Leftrightarrow$ logische 0
blauer Kobold $\Leftrightarrow$ logisches NOT
roter Kobold $\Leftrightarrow$ logisches OR
gelber Kobold $\Leftrightarrow$ logisches AND
Nun stellt man "bottom-up" eine Aussagenlogische Formel auf:
$(A \vee \overline{BC}) \wedge (\overline{A} \vee \overline{B} \vee \overline{E}) \wedge (A \vee B \vee D) \wedge (C \vee D) \wedge (\overline{AC}) \wedge (A \vee \overline{E}) \wedge (\overline{D\overline{E}})$
Durch Umformen mit den bekannten Formeln kommt man zu folgendem Ergebnis:
$A\overline{BC}DE$
Somit muss das Christkind in Trichter A, D und E Schokolade und in Trichter B und C Schlangengift schütten.

Grammatiken und Chomsky-Hierarchie

Definition: Grammatik

Eine **Grammatik** G besteht aus vier Komponenten:

1. einem endlichen Alphabet Σ (**Terminalalphabet**)
2. einer endlichen Menge V mit $V \cap \Sigma = \emptyset$ von Variablen (**Nichtterminale**)
3. einem **Startsymbol** $S \in V$
4. einer endlichen Menge von **Ableitungsregeln** R (Produktionen).
   Dabei ist eine Ableitungsregel ein Paar (l, r) mit $l \in (V \cup \Sigma)^+$ und $r \in (V \cup \Sigma)^*$. Statt (l, r) schreiben wir auch $l \to r$.

Wir schreiben $w \longrightarrow z$, wenn w durch eine Anwendung einer Ableitungsregel in z verwandelt wird und $w \overset{*}{\longrightarrow} z$, wenn w durch eine Anwendung von mehreren Ableitungsregeln in z verwandelt wird.

Wir schreiben $w \rightarrow z$, wenn w durch eine Anwendung einer Ableitungsregel in z verwandelt wird und $w \xrightarrow{*} z$, wenn w durch eine Anwendung von mehreren Ableitungsregeln in z verwandelt wird.

Definition: erzeugte Sprache

Die von einer Grammatik G **erzeugte Sprache** $L(G)$ ist die Menge aller Wöerter $z \in \Sigma^*$, für die $S \xrightarrow{*} z$ gilt.

Aufgabe 1

Gib eine Grammatik G an, mit der sich die korrekt geklammerten aussagenlogischen Formeln darstellen lassen.
Überprüfe ob $(\neg((v \vee v) \wedge (\neg v))) \in L(G)$.

Aufgabe 1

Gib eine Grammatik G an, mit der sich die korrekt geklammerten aussagenlogischen Formeln darstellen lassen.
Überprüfe ob $(\neg((v \vee v) \wedge (\neg v))) \in L(G)$.

Lösung:
$G = (\{A, B\}, \{(,), \vee, \wedge, \neg, 0, 1, v\}, B, R)$ mit $R = \{$
$B \to (B \vee B)$
$B \to (B \wedge B)$
$B \to (\neg B)$
$B \to A$
$A \to v$
$A \to 0$
$A \to 1\}$
Nicht die kürzeste Grammatik, aber die intuitiv einfachste!

Definition: Chomsky-Hierarchie

- Grammatiken ohne weitere Einschränkungen heissen Grammatiken vom **Typ 0**.

Definition: Chomsky-Hierarchie

- Grammatiken ohne weitere Einschränkungen heissen Grammatiken vom **Typ 0**.
- Grammatiken, bei denen alle Ableitungsregeln die Form

  $u \to v$ mit $u \in V^+, v \in ((V \cup \Sigma) \backslash \{S\})^+$ und $|u| \leq |v|$, oder $S \to \varepsilon$

  haben, heissen **kontextsensitiv** od. Grammatiken vom **Typ 1**.

Definition: Chomsky-Hierarchie

- Grammatiken ohne weitere Einschränkungen heissen Grammatiken vom **Typ 0**.
- Grammatiken, bei denen alle Ableitungsregeln die Form

  $u \to v$ mit $u \in V^+, v \in ((V \cup \Sigma) \backslash \{S\})^+$ und $|u| \leq |v|$, oder $S \to \varepsilon$

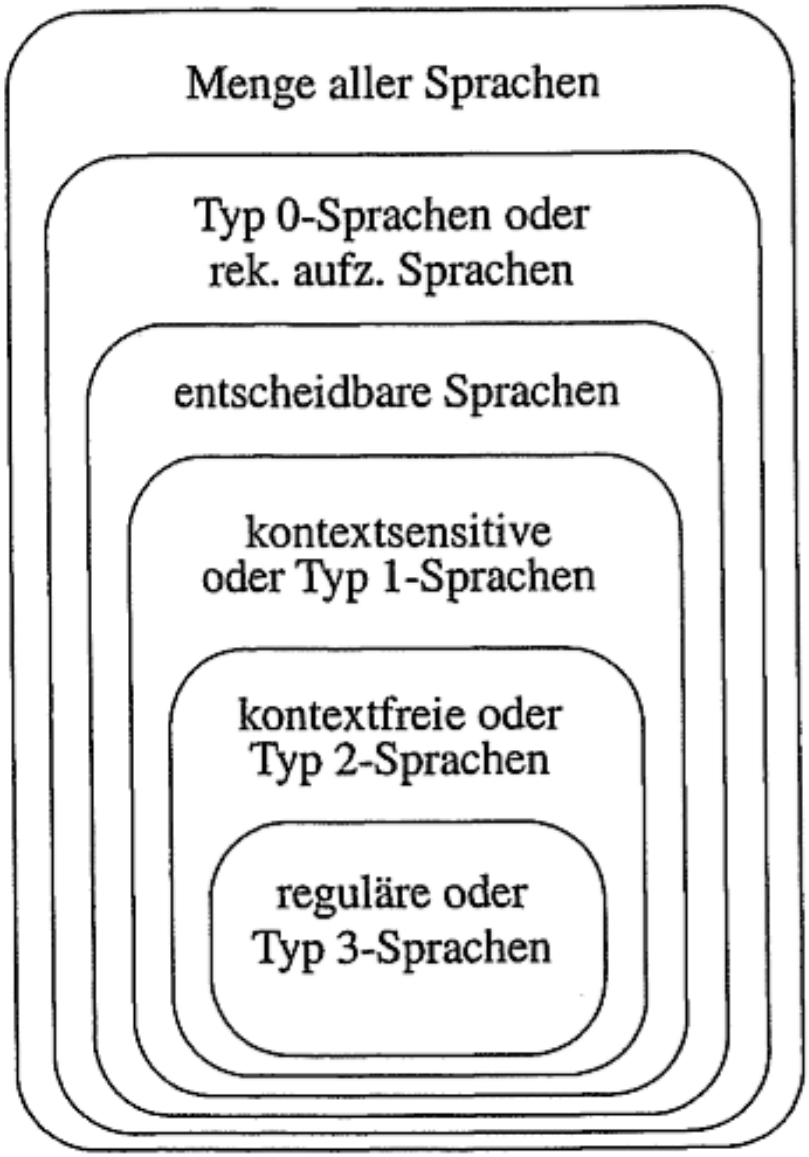
  haben, heissen **kontextsensitiv** od. Grammatiken vom **Typ 1**.
- Grammatiken, bei denen alle Ableitungsregeln die Form $A \to v$ mit $A \in V$ und $v \in (V \cup \Sigma)^*$ haben, heissen **kontextfrei** oder Grammatiken vom **Typ 2**.

Definition: Chomsky-Hierarchie

- Grammatiken ohne weitere Einschränkungen heissen Grammatiken vom **Typ 0**.
- Grammatiken, bei denen alle Ableitungsregeln die Form

  $u \to v$ mit $u \in V^+, v \in ((V \cup \Sigma) \backslash \{S\})^+$ und $|u| \leq |v|$, oder $S \to \varepsilon$

  haben, heissen **kontextsensitiv** od. Grammatiken vom **Typ 1**.
- Grammatiken, bei denen alle Ableitungsregeln die Form $A \to v$ mit $A \in V$ und $v \in (V \cup \Sigma)^*$ haben, heissen **kontextfrei** oder Grammatiken vom **Typ 2**.
- Grammatiken, bei denen alle Ableitungsregeln die Form $A \to v$ mit $A \in V$ und $v = \varepsilon$ oder $v = aB$ mit $a \in \Sigma, B \in V$ haben, heissen **rechtslinear** oder Grammatiken vom **Typ 3**.


Menge aller Sprachen
Typ 0-Sprachen oder
rek. aufz. Sprachen
entscheidbare Sprachen
kontextsensitive
oder Typ 1-Sprachen
kontextfreie oder
Typ 2-Sprachen
reguläre oder
Typ 3-Sprachen

Satz

Falls Sprache L rekursiv aufzählbar / semi-entscheidbar ist, so gibt es eine Chomsky-0-Grammatik mit $L(G) = L$.
Die von Chomsky-0-Grammatiken G erzeugten Sprachen sind rekursiv aufzählbar.

Satz

Falls Sprache L rekursiv aufzählbar / semi-entscheidbar ist, so gibt es eine Chomsky-0-Grammatik mit $L(G) = L$.
Die von Chomsky-0-Grammatiken G erzeugten Sprachen sind rekursiv aufzählbar.

Warum?
Kurz: L rekursiv aufzählbar $\rightarrow \exists$ DTM M, die die Wörter $w \in L$ akzeptiert.

Beispiel:

Satz

Falls Sprache L rekursiv aufzählbar / semi-entscheidbar ist, so gibt es eine Chomsky-0-Grammatik mit $L(G) = L$.
Die von Chomsky-0-Grammatiken G erzeugten Sprachen sind rekursiv aufzählbar.

Warum?
Kurz: L rekursiv aufzählbar $\rightarrow \exists$ DTM M, die die Wörter $w \in L$ akzeptiert.

Beispiel:
Halteproblem.

Satz

Die Klasse der von endichen Automaten akzeptierten Sprachen ist genau die Klasse der von Chomsky-3-Grammatiken erzeugten Sprachen.

Aus endlichem Automaten eine Typ 3 Grammatik konstruieren (Satz 5.5):

Aus endlichem Automaten eine Typ 3 Grammatik konstruieren (Satz 5.5):

Sei A_L ein (deterministischer) endlicher Automat, der die Sprache $L \subseteq \Sigma^*$ akzeptiert, $A_L = (Q, \Sigma, \delta, q_0, F)$. Dann kann man die zugehörige Grammatik G_L wie folgt konstruieren:

$V := Q$

$S := q_0$

R enthält die Regel $q \to \varepsilon$ für alle $q \in F$ und die Regel $q \to aq'$, falls $\delta(q, a) = q'$

Für $w = w_1 ... w_n \in L$ durchläuft A_L genau die Zustände $q_0, q_1, ..., q_n \in Q$ mit $q_n \in F$. Dann gilt:
$q_0 \to w_1 q_1 \to w_1 w_2 q_2 \to ... \to w q_n \to w$

Aufgabe 2

Wir betrachten den endlichen Automaten über dem Alphabet $\Sigma = \{0, 1\}$ mit den Zuständen $\{A, B, C, D\}$. Startzustand ist A, einziger Endzustand D. Übergangstabelle ist

δ	A	B	C	D
0	A,B	D	A,D	B
1	C	A	A	C

Sei L die Sprache die der Automat akzeptiert. Gebe eine Typ-k Grammatik mit maximalem k an, die L erzeugt.

Aufgabe 3

Gegeben ist die Grammatik $G = (\Sigma, V, S, R)$ mit $\Sigma = \{a, b, c\}$, $V = \{S, A\}$ und $R = \{S \to AA, A \to AAA|abA|aAb|bAa|baA|c\}$

1. Gebe den maximalen Typ von G in der Chomsky-Hierachie an und begründe deine Antwort.
2. Bestimme alle Wörter aus $L(G)$, die sich durch Anwendung von maximal vier Ableitungsregeln ergeben.
3. Ist das Wort $w = acbbca$ in $L(G)$ enthalten?
4. Zeige, dass die Sprache $L(\alpha)$, die durch den regulären Ausdruck

$$\alpha := ((ab \cup ba)^* c (ab \cup ba)^* c)^+$$

gegeben ist, in $L(G)$ enthalten ist.

Aus Typ 3 Grammatik einen nichtdeterministischen endlichen Automaten konstruieren (Satz 5.5).

Aus Typ 3 Grammatik einen nichtdeterministischen endlichen Automaten konstruieren (Satz 5.5).

Sei eine Chomsky-3-Grammatik G_L gegeben. Wir entwerfen einen NEA $A_L := (Q, \Sigma, \delta, q_0, F)$, der L akzeptiert. Und zwar auf folgende Art:

$$Q := V$$
$$q_0 := S$$
$$F := \{A \in V | (A \to \varepsilon) \in R\}$$
$$\delta(A, a) := \{B | (A \to aB) \in R\}$$

Für $w = w_1 ... w_n \in L$ hat die Ableitung von w mittels G_L das Aussehen:
$S \to w_1 A_1 \to w_1 w_2 A_2 \to ... \to w A_n \to w$

Aufgabe 4

Gegeben sie das Alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -\}$.
Betrachte die regulären Ausdrücke z,n,g:

$$\begin{aligned} z &= 0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9 \\ n &= zz^* \\ g &= (+ \cup - \cup \epsilon)n \end{aligned}$$

- Gebe eine rechtlineare Grammatik über dem Alphabet Σ an, deren erzeugende Sprache gerade $L(z)$, $L(n)$ beziehungsweise $L(g)$ ist.
- Konstruiere jeweils den DEA, der $L(z)$, $L(n)$ und $L(g)$ erkennt und sich aus der Konstruktionsvorschrift in Satz 5.5 der Vorlesung ergibt.

Aufgabe 5

Zeige, dass die Grammatik $G = (\{a, b\}, \{S\}, S, R)$ mit den Produktionen $R = \{S \rightarrow ab, S \rightarrow aSb\}$ die Sprache $\{a^n b^n | n \geq 1\}$ erzeugt.

Aufgabe 5

Zeige, dass die Grammatik $G = (\{a, b\}, \{S\}, S, R)$ mit den Produktionen $R = \{S \to ab, S \to aSb\}$ die Sprache $\{a^n b^n | n \geq 1\}$ erzeugt.
Hinweis: Induktion über die Ableitungslänge.

Reflexion

Was haben wir heute gelernt?

Reflexion

Was haben wir heute gelernt?

- Übeblick über die Chomsky-Hierarchie

Reflexion

Was haben wir heute gelernt?

- Übeblick über die Chomsky-Hierarchie
- Chomsky-3-Sprachen

Noch Fragen?

Vorschau

Vorschau

- Chomsky-1-Grammatiken

Vorschau

- Chomsky-1-Grammatiken
- Chomsky-2-Grammatiken

Bis zum nächsten Mal