Informatik III - Tutorium IX & X (SR -107)

Tut Nr. 6 – Üb5, Berechenbarkeit

David Münch

Universität Karlsruhe (TH)
Institut für Informatik
IAKS Beth

3. Dezember 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 Donnerstags.

Was wollen wir heute erreichen?

Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen

Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen
- WHILE-Programme

Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen
- WHILE-Programme
- LOOP-Programme

Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen
- WHILE-Programme
- LOOP-Programme
- GOTO-Grogramme

Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen
- WHILE-Programme
- LOOP-Programme
- GOTO-Grogramme
- Ackermannfunktion

Aufgabe 1

Gegeben sei folgende Grammatik $\mathcal{G} = \{\mathcal{A}, \mathcal{V}, S, \mathcal{P}\}$ mit den Terminalsymbolen $\mathcal{A} = \{a, b, c, d\}$, den Nichterminalen $\mathcal{V} = \{S, X, Y, Z\}$, dem Startzeichen $S \in \mathcal{V}$ und den Produktionen

$$\begin{aligned} \mathcal{P} = \{S &\rightarrow AbC \\ A &\rightarrow aaA|aa \\ C &\rightarrow ccC|c \\ aaA &\rightarrow dAc\}\,. \end{aligned}$$

Welche Sprache $\mathcal{L}_\mathcal{G}$ erzeugt die Grammatik $\mathcal{G}$?

Aufgabe 1

Gegeben sei folgende Grammatik $\mathcal{G} = \{\mathcal{A}, \mathcal{V}, S, \mathcal{P}\}$ mit den Terminalsymbolen $\mathcal{A} = \{a, b, c, d\}$, den Nichterminalen $\mathcal{V} = \{S, X, Y, Z\}$, dem Startzeichen $S \in \mathcal{V}$ und den Produktionen

$$\begin{aligned} \mathcal{P} = \{S &\rightarrow AbC \\ A &\rightarrow aaA|aa \\ C &\rightarrow ccC|c \\ aaA &\rightarrow dAc\}\,. \end{aligned}$$

Welche Sprache $\mathcal{L}_\mathcal{G}$ erzeugt die Grammatik $\mathcal{G}$?

Die Grammatik $\mathcal{G}$ erzeugt die Sprache

$$\mathcal{L}_\mathcal{G} = \left\{ (\prod_{i=1}^{s} a^{2r_i} d) a^{2+2n} c^s b c^{2t+1} | r_1, \ldots, r_s, s, n, t \in \mathbb{N}_0 \right\}.$$

Aufgabe 1

Gegeben sei folgende Grammatik $\mathcal{G} = \{\mathcal{A}, \mathcal{V}, S, \mathcal{P}\}$ mit den Terminalsymbolen $\mathcal{A} = \{a, b, c, d\}$, den Nichterminalen $\mathcal{V} = \{S, X, Y, Z\}$, dem Startzeichen $S \in \mathcal{V}$ und den Produktionen

$$\begin{aligned} \mathcal{P} = \{S &\rightarrow AbC \\ A &\rightarrow aaA|aa \\ C &\rightarrow ccC|c \\ aaA &\rightarrow dAc\}\,. \end{aligned}$$

Geben Sie für das Wort $w = aadaabccc$ eine Ableitung an. Geben Sie ein PCP an, das zum Wortproblem $w \in \mathcal{L}_{\mathcal{G}}$ äquivalent ist.

Aufgabe 1

Gegeben sei folgende Grammatik $\mathcal{G} = \{\mathcal{A}, \mathcal{V}, S, \mathcal{P}\}$ mit den Terminalsymbolen $\mathcal{A} = \{a, b, c, d\}$, den Nichterminalen $\mathcal{V} = \{S, X, Y, Z\}$, dem Startzeichen $S \in \mathcal{V}$ und den Produktionen

$$\begin{aligned} \mathcal{P} = \{S &\rightarrow AbC \\ A &\rightarrow aaA|aa \\ C &\rightarrow ccC|c \\ aaA &\rightarrow dAc\}\,. \end{aligned}$$

Geben Sie für das Wort $w = aadaabccc$ eine Ableitung an. Geben Sie ein PCP an, das zum Wortproblem $w \in \mathcal{L}_{\mathcal{G}}$ äquivalent ist.

Eine mögliche Ableitung ist: $S \Rightarrow AbC \Rightarrow aaAbC \Rightarrow aaaaAbC \Rightarrow aadAcbC \Rightarrow aadaacbC \Rightarrow aadaacbccC \Rightarrow aadaacbccc$

Aufgabe 1

Gegeben sei folgende Grammatik $\mathcal{G} = \{\mathcal{A}, \mathcal{V}, S, \mathcal{P}\}$ mit den Terminalsymbolen $\mathcal{A} = \{a, b, c, d\}$, den Nichterminalen $\mathcal{V} = \{S, X, Y, Z\}$, dem Startzeichen $S \in \mathcal{V}$ und den Produktionen

$$\begin{aligned} \mathcal{P} = \{S &\rightarrow AbC \\ A &\rightarrow aaA|aa \\ C &\rightarrow ccC|c \\ aaA &\rightarrow dAc\}\,. \end{aligned}$$

Das äquivalente Postsystem lautet:

$$\binom{A'b'C'}{S}, \binom{AbC}{S'}, \binom{a'a'A'}{A}, \binom{aaA}{A'}, \binom{a'a'}{A}, \binom{aa}{A'}, \binom{c'c'C'}{C}, \binom{ccC}{C'}, \binom{c'}{C}, \binom{c}{C'},$$
$$\binom{d'A'c'}{aaA}, \binom{dAc}{a'a'A'}, \binom{a'}{a}, \binom{a}{a'}, \binom{b'}{b}, \binom{b}{b'}, \binom{c'}{c}, \binom{c}{c'}, \binom{d'}{d}, \binom{d}{d'},$$
$$\binom{*'}{*}, \binom{*}{*'}, \binom{S*'A'b'C'}{S}, \binom{\lambda}{*aadaacbccc}$$

Aufgabe 2

Gegeben sei die Grammatik $\mathcal{G} = (\mathcal{A}, \mathcal{V}, S, \mathcal{P})$, $\mathcal{A} = \{a, b\}$ mit den Produktionen

$$\begin{aligned} \mathcal{P} = \{ \quad & S \to Aa|Aab|aB|a, \\ & A \to aAb|ba, \\ & B \to bS|Aa\} \end{aligned}$$

1. Formulieren sie für die Grammatik $\mathcal{G}$ eine Abbildung $g : \left(2^{\mathcal{A}^*}\right)^3 \to \left(2^{\mathcal{A}^*}\right)^3$

$$g\begin{pmatrix} \mathcal{L}_S \\ \mathcal{L}_A \\ \mathcal{L}_B \end{pmatrix} = \begin{pmatrix} \mathcal{L}_A(a + ab) + a\mathcal{L}_B + a \\ a\mathcal{L}_A b + ba \\ b\mathcal{L}_S + \mathcal{L}_A a \end{pmatrix}$$

2. Berechnen sie den minimalen Fixpunkt der Abbildung g

Definition:

- Eine Sprache $L \subseteq \Sigma^*$ heisst **rekursiv** oder **entscheidbar**, wenn es eine TM gibt, die auf allen Eingaben stoppt und eine Eingabe w genau dann akzeptiert, wenn $w \in L$ gilt.

Definition:

- Eine Sprache $L \subseteq \Sigma^*$ heisst **rekursiv** oder **entscheidbar**, wenn es eine TM gibt, die auf allen Eingaben stoppt und eine Eingabe w genau dann akzeptiert, wenn $w \in L$ gilt.
- Eine Sprache $L \subseteq \Sigma^*$ heisst **rekursiv-aufzählbar** oder **semi-entscheidbar**, wenn es eine TM gibt, die genau die Eingaben w akzeptiert für die $w \in L$. (Das Verhalten der TM für $w \notin L$ ist nicht defniniert.)

Definition:

- Eine Funktion $f : \Sigma^* \rightarrow \Gamma^*$ heisst **berechenbar** oder **totalrekursiv**, wenn es eine TM gibt, die bei Eingabe von $w \in \Sigma^*$ den Funktionswert $f(w) \in \Gamma^*$ ausgibt.

Definition:

- Eine Funktion $f : \Sigma^* \to \Gamma^*$ heisst **berechenbar** oder **totalrekursiv**, wenn es eine TM gibt, die bei Eingabe von $w \in \Sigma^*$ den Funktionswert $f(w) \in \Gamma^*$ ausgibt.
- Eine TM realisiert eine Funktion $f : \Sigma^* \to \Gamma^*$, falls gilt:

$$f(w) = \begin{cases} \text{Ausgabe der TM,} & \text{wenn sie bei Eingabe } w \text{ stoppt.} \\ \text{undefiniert,} & \text{sonst} \end{cases}$$

Definition:

- Eine Sprache $L \subseteq \Sigma^*$ ist **entscheidbar** $\Leftrightarrow$ ihre charakterische Funktion χ_L berechenbar ist, wobei gilt:

$$\chi_L : \Sigma^* \to \{0,1\} \text{ mit } \chi_L(w) = \begin{cases} 1, & \text{falls } w \in L \\ 0, & \text{sonst.} \end{cases}$$

Definition:

- Eine Sprache $L \subseteq \Sigma^*$ ist **entscheidbar** $\Leftrightarrow$ ihre charakterische Funktion χ_L berechenbar ist, wobei gilt:

$$\chi_L : \Sigma^* \to \{0,1\} \text{ mit } \chi_L(w) = \begin{cases} 1, & \text{falls } w \in L \\ 0, & \text{sonst.} \end{cases}$$

- Eine Sprache $L \subseteq \Sigma^*$ ist **semi-entscheidbar** $\Leftrightarrow$ Funktion χ_L^* berechenbar ist, wobei gilt:

$$\chi_L^*(w) = \begin{cases} 1, & \text{falls } w \in L \\ \text{undefiniert}, & \text{sonst.} \end{cases}$$

Satz von Rice

Sei R die Menge der von Turing-Maschinen berechenbaren Funktionen und S eine nicht-triviale Teilmenge von R ($\emptyset \neq S \neq R$). Dann ist die Sprache:
$L(S) := \{\langle M \rangle \mid M \text{ berechnet eine Funktion aus } S\}$
nicht entscheidbar.

Aufgabe 1

Zeige, dass die Menge der Turing-Maschinencodes von TMs, die alle aus Palindromen bestehenden Eingaben (möglicherweise zusammen mit anderen Eingaben) akzeptieren, unentscheidbar ist.

Aufgabe 1

Zeige, dass die Menge der Turing-Maschinencodes von TMs, die alle aus Palindromen bestehenden Eingaben (möglicherweise zusammen mit anderen Eingaben) akzeptieren, unentscheidbar ist.

Lösung: Die Eigenschaft "aus Palindromen bestehenden Eingaben" ist nicht trivial, weil einige Sprachen sie erfüllen und andere Sprachen nicht. Somit nach dem Satz von Rice unentscheidbar.

Churche These:

Die durch die formale Definition der Turing-Berechenbarkeit erfasste Klasse von Funktionen stimmt genau mit der Klasse der im intuitiven Sinne berechenbaren Funktionen überein.

Zu berechnen

Folglich kann Funktion f nach Church'scher These durch TM berechnet werden:

$$f(x,y,z) := \begin{cases} (x^2, y^2) & ,z = x * y \text{ und } z \neq 1 \\ \perp & ,z = 1 \\ (x^4, 17y) & ,sonst \end{cases}$$

Zu berechnen

Folglich kann Funktion f nach Church'scher These durch TM berechnet werden:

$$f(x,y,z) := \begin{cases} (x^2, y^2) & ,z = x * y \text{ und } z \neq 1 \\ \perp & ,z = 1 \\ (x^4, 17y) & ,sonst \end{cases}$$

Offensichtlich mit TM sehr aufwendig $\rightarrow$ suche ein anderes Rechenmodell, dass einer TM entspricht.

1. Rechenmodell: while-Programme

Besteht aus:

- Variablen $x_1, \ldots, x_m$
- Konstanten $0, 1, 2, \ldots$
- Zeichen $;, :=, +, -$
- Schlüsselworten **while**, **do**, **end**

Makros

Ein Makro ist eine Kurznotation für ein **while**-Programm.
Beispiel: **if**. . . **then**. . . **else**

Weitere Makros

$x := y$,
$x := y + z$,
$x := y - z$, (falls $y < z$, dann $x = 0$)
$x := y^z$,
$x := y \mod z$,
$x := y$ div z,
$x := \log_2 y$, (ganzzahliger Anteil von $\log_2 y$)
$x := n$,
$x := (x * y) * n$,
$x := m * n$.

Aufgabe

Übersetze folgende Funktion f in ein **while**-Programm:

$$f(x,y,z) := \begin{cases} (x^2, y^2) & ,z = x * y \text{ und } z \neq 1 \\ \bot & ,z = 1 \\ (x^4, 17y) & ,sonst \end{cases}$$

Lösung

if z=1 **then** **while** $z \neq 0$ **do** $z := z + 1$ **end**

else **if** $z := x * y;$ **then** $u := 2;\ x := x^u;\ y := y^u;$

else $u := 4;\ v := 17;$
$x := x^u; y := y * v;$
end

end

2. Rechenmodell: goto-Programme

Folge nummerierter Anweisungen:

1. A_1
2. A_2

   ...

n. A_n

Die Anweisungen bestehen aus

- Zuweisung: $x := y + z, x := y - z$
- unbedingten Sprüngen: **goto** k
- bedingten Sprüngen: **if** $x = y$ **then goto** k
- Stopp-Anweisungen: **stop**

Aufgabe

Übersetze folgende Funktion f in ein **goto**-Programm:

$$f(x,y,z) := \begin{cases} (x^2, y^2) & ,z = x * y \text{ und } z \neq 1 \\ \bot & ,z = 1 \\ (x^4, 17y) & ,sonst \end{cases}$$

Lösung

1. **if** $z = 1$ **then goto** 1

2. **if** $z \neq x * y$ **then goto** 3
   $u := 2$
   $x = x^u$
   $y = y^u$
   goto 4

3. $u := 4, \quad v := 17$
   $x := x^u, \quad y := y * w$

4. **stop**

3. Rechenmodell: loop-Programme

Ein **loop**-Programm ist ein **while**-Programme, der Form

loop x **do** P **end,**

so dass die Anzahl der Durchläufe durch x begrenzt ist.

Aufgabe

Übersetze folgende Funktion f in ein **loop**-Programm:

$$f(x,y,z) := \begin{cases} (x^2, y^2) & ,z = x * y \text{ und } z \neq 1 \\ \bot & ,z = 1 \\ (x^4, 17y) & ,sonst \end{cases}$$

Lösung

Es kann kein entsprechendes Programm angegeben werden, da **loop**-Programme nur *total-berechenbare* Funktionen berechnen können.

Lösung

Es kann kein entsprechendes Programm angegeben werden, da **loop**-Programme nur *total-berechenbare* Funktionen berechnen können.

Definition: total berechenbar

Eine Funktion heißt *total berechenbar*, wenn sie für alle Eingaben definiert ist.

Lösung

Es kann kein entsprechendes Programm angegeben werden, da **loop**-Programme nur *total-berechenbare* Funktionen berechnen können.

Definition: total berechenbar

Eine Funktion heißt *total berechenbar*, wenn sie für alle Eingaben definiert ist.

$\longrightarrow$ In welcher Beziehung stehen die 3 Rechenmodelle?

$$\mathbf{WHILE} \iff \mathbf{GOTO} \iff \mathbf{TM}$$
$$\Uparrow$$
$$\mathbf{LOOP}$$

WHILE $\Longleftrightarrow$ **GOTO** $\Longleftrightarrow$ **TM**

$\Uparrow$

LOOP

Turingmaschine, **while**-Programm und **goto**-Programm sind äquivalent. Beweisidee: gib jeweils eine entsprechende Überführung an.

WHILE $\Longleftrightarrow$ **GOTO** $\Longleftrightarrow$ **TM**

$\Uparrow$

LOOP

Turingmaschine, **while**-Programm und **goto**-Programm sind äquivalent. Beweisidee: gib jeweils eine entsprechende Überführung an.

loop-Programme sind nicht so mächtig (Intutition: Die Anzahl der Durchläufe muss vorher bekannt sein).
Beweis mit Hilfe der Ackermannfunktion.

Definition: Ackermannfunktion nach Péter

$$\mathbb{N}^2 \to \mathbb{N}$$
$$a(0, m) = m + 1$$
$$a(n + 1, 0) = a(n, 1)$$
$$a(n + 1, m + 1) = a(n, a(n + 1, m))$$

Hinweis: Die Ackermannfunktion ist eine extrem schnell wachsende Funktion.

Aufgabe

Zeige für die Ackermannfunktion:

a) $y < a(x, y)$

b) $a(x, y) < a(x, y + 1)$

Lösung:

a) Zu Zeigen ist $y < a(x, y)$ für alle x und y und zwar durch Induktion über x

(i) $a(0, y) = y + 1 > y$ für alle y

(ii) Induktionsvoraussetzung ist $y < a(x, y)$ für ein (beliebiges) x und alle y. Den Nachweis von $y < a(x + 1, y)$ für alle y führen wir durch Induktion über y:

1. $y = 0 : \quad a(x + 1, 0) = a(x, 1) > 1 > 0$
2. Induktionsvoraussetzung ist die Gültigkeit für $x + 1$ und ein y. Nachzuweisen (Induktionsschritt) ist die Gültigkeit für $y + 1$.

$$\begin{aligned} a(x + 1, y + 1) &= a(x, a(x + 1, y)) \\ &> a(x + 1, y) && \text{Induktionsvoraussetzung für } x \\ &> y && \text{Induktionsvoraussetzung für } y \end{aligned}$$

Aus $a(x + 1, y) > y$ folgt $a(x + 1, y) \geq y + 1$ und daraus die Behauptung

Lösung:

b) Fallunterscheidung für x

1. Für $x = 0$ folgt die Behauptung aus $a(0, y + 1) = y + 2$ und $a(0, y) = y + 1$.
2. Für $x > 0$ ist $a(x, y + 1) = a(x - 1, a(x, y)) > a(x, y)$ wegen Teilaufgabe a.

Dank

Dank an Benjamin Niedermann für LaTeX Quellcode.

Reflexion

Was haben wir heute gelernt?

Reflexion

Was haben wir heute gelernt?

- Theorie zur Berechenbarkeit

Reflexion

Was haben wir heute gelernt?

- Theorie zur Berechenbarkeit
- Satz von Rice anwenden

Reflexion

Was haben wir heute gelernt?

- Theorie zur Berechenbarkeit
- Satz von Rice anwenden
- WHILE, LOOP, GOTO

Reflexion

Was haben wir heute gelernt?

- Theorie zur Berechenbarkeit
- Satz von Rice anwenden
- WHILE, LOOP, GOTO
- Ackermannfunktion

Noch Fragen?

Vorschau

Vorschau

- Unentscheidbare Probleme

Vorschau

- Unentscheidbare Probleme
- ...?

Bis zum nächsten Mal