Technische Universität München
Fakultät für Informatik
Lehrstuhl Informatik XIV
Prof. Dr. Ernst W. Mayr
Dr. Werner Meixner

Sommersemester 2015
Lösungsblatt 10
6. Juli 2015

Theoretische Informatik

Hausaufgabe 1 (4 Punkte)

Seien K_1 eine kontextfreie Sprache und K_2, K_3 deterministische kontextfreie Sprachen über Σ. Zeigen Sie:

1. Für $\overline{K_2 \cap K_3}$ ist das Leerheitsproblem entscheidbar.
2. Für $\overline{K_1 \cap K_2}$ ist das Wortproblem entscheidbar.

Lösung

1. Nach Vorlesung, Abschnitt 6.4 Entscheidbarkeit, ist das Leerheitsproblem für kontextfreie Sprachen entscheidbar. Also genügt es zu zeigen, dass $\overline{K_2 \cap K_3}$ kontextfrei ist. Es gilt
$$\overline{K_2 \cap K_3} = \overline{K_2} \cup \overline{K_3}\,.$$
Nach Vorlesung, Abschnitt 6.3 Abschlusseigenschaften, sind das Komplement von DCFL Sprachen wiederum DCFL Sprachen, mithin sind $\overline{K_2}$ und $\overline{K_3}$ kontextfreie Sprachen. Die Vereinigung $\overline{K_2} \cup \overline{K_3}$ kontextfreier Sprachen $\overline{K_2}$ und $\overline{K_3}$ ist wiederum kontextfrei. (2P)
2. Nach Vorlesung, Abschnitt 6.4 Entscheidbarkeit, ist das Wortproblem für kontextsensitive Sprachen entscheidbar. Also genügt es zu zeigen, dass $\overline{K_1 \cap K_2}$ kontextsensitiv ist.
Da K_1 und K_2 insbesondere kontextsensitiv sind, wenden wir lediglich die in Abschnitt 6.4 der Vorlesung genannte Tatsache an, dass sowohl der Durchschnitt, als auch das Komplement kontextsensitiver Sprachen wieder kontextsensitiv ist. (2P)

Hausaufgabe 2 (4 Punkte)

1. Geben Sie eine deterministische Turingmaschine $T = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ an, die für ein Eingabewort $w \in \{0,1\}^+$ eine Berechnung durchführt, so dass am Ende der Berechnung der Kopf am Anfang des Wortes w^R auf dem sonst leeren Band steht. (Erinnerung: w^R ist das gespiegelte Wort zu w.)

   Beschreiben Sie zunächst informell Ihre Lösungsidee!
2. Zeigen Sie durch Angabe einer geeigneten Konfigurationsfolge, dass Ihre Turingmaschine T die Eingabe 10 korrekt verarbeitet.

Lösung

1. Konstruktionsbeschreibung: Ein Zeichen x wird gelesen, es wird $\widehat{x}$ als Markierung geschrieben, im Zustand q^x an den Wortanfang gefahren und dort $\overline{x}$ geschrieben. Dann erfolgt Rücklauf an den Anfang des unmarkierten Wortes und Wiederholung, so lange noch unmarkierte x vorhanden sind. Andernfalls wird an den Anfang des geschriebenen Wortes gefahren und dabei alle $\widehat{x}$ gelöscht und alle $\overline{x}$ in x umgewandelt.

   Wir wählen $\Gamma = \Sigma \cup \widehat{\Sigma} \cup \overline{\Sigma} \cup \{\square\}$ mit $\Sigma = \{0,1\}$, $\widehat{\Sigma} = \{\widehat{x} \mid x \in \Sigma\}$ und $\overline{\Sigma} = \{\overline{x} \mid x \in \Sigma\}$. Sei $Q = \{q_0, q_R, q_L, q_e\} \cup \{q^x \mid x \in \Sigma\}$ mit q_0 als Start- und q_e als Endzustand. Die Markierung $\widehat{\ }$ verwenden wir um das Zeichen zu markieren, welches gerade kopiert wurde. Die Zeichen der Kopie auf der linken Seite von w versehen wir zunächst mit einem $\overline{\ }$, um sie von den Zeichen des Originalwortes unterscheiden zu können.

Übergang	Bereich	Kommentar
$\delta(q_0, x) = (q^x, \widehat{x}, L)$	$x \in \Sigma$	Speichern erster Buchstabe.
$\delta(q^x, y) = (q^x, y, L)$	$y \in \widehat{\Sigma} \cup \overline{\Sigma}$	Gehe links zum Ausgabeanfang.
$\delta(q^x, \square) = (q_R, \overline{x}, N)$		Schreibe x zum Ausgabeanfang.
$\delta(q_R, y) = (q_R, y, R)$	$y \in \widehat{\Sigma} \cup \overline{\Sigma}$	Laufe nach rechts zum ersten x.
$\delta(q_R, x) = (q_0, x, N)$	$x \in \Sigma$	Wiederholung.
$\delta(q_R, \square) = (q_L, \square, L)$	$y \in \widehat{\Sigma}$	Beginn Linkslauf.
$\delta(q_L, y) = (q_L, \square, L)$	$y \in \widehat{\Sigma}$	Löschung Eingabe.
$\delta(q_L, \overline{x}) = (q_L, x, L)$	$x \in \Sigma$	Markierung entfernen.
$\delta(q_L, \square) = (q_e, \square, R)$		Stop. (2P)

2. $(\epsilon, q_0, 10) \to_T (\epsilon, q^1, \square\widehat{1}0) \to_T (\epsilon, q_R, \overline{1\widehat{1}}0) \to_T (\overline{1}, q_R, \widehat{1}0) \to_T (\overline{1\widehat{1}}, q_R, 0) \to_T$
   $(\overline{1\widehat{1}}, q_0, 0) \to_T (\overline{1\widehat{1}}, q^0, \widehat{0}) \to_T (\overline{1}, q^0, \widehat{10}) \to_T (\epsilon, q^0, \overline{1\widehat{10}}) \to_T (\epsilon, q^0, \square\overline{1\widehat{10}}) \to_T$
   $(\epsilon, q_R, \overline{01\widehat{10}}) \to_T (\overline{0}, q_R, \overline{1\widehat{10}}) \to_T \quad \ldots \quad (\overline{01\widehat{10}}, q_R, \epsilon) \to_T$
   $(\overline{01\widehat{1}}, q_L, \widehat{0}\square) \to_T (\overline{01}, q_L, \widehat{1}\square\square) \to_T (\overline{0}, q_L, \overline{1}\square\square\square) \to_T (\epsilon, q_L, \overline{0}1\square\square\square) \to_T$
   $(\epsilon, q_L, \square 01\square\square\square) \to_T (\square, q_e, 01\square\square\square)$. (2P)

Hausaufgabe 3 (4 Punkte)

Eine deterministische Turingmaschine $T = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ nennen wir eingeschränkt speicherfähig, wenn T bei der Abänderung einer Feldinschrift den Kopf nicht bewegt. Formal wird dies durch die folgende Eigenschaft E der Übergangsfunktion δ von T definiert, i.Z. $T \in E$:

$$\text{(E)}\ \forall x, y \in \Gamma, p, q \in Q, d \in \{L, R, N\} F: \quad (\, x \neq y \ \wedge\ \delta(p, x) = (q, y, d)\,) \implies d = N\,.$$

1. Definieren Sie eine eingeschränkt speicherfähige Turingmaschine $T \in E$, die für jede nichtleere Eingabe $w = x_1 x_2 \ldots x_n$ mit Dezimalziffern $x_i \in \Sigma = \{0, 1, 2, \ldots, 9\}$ eine Berechnung durchführt, so dass bei Terminierung das Wort $w' = x_1 x_1 \ldots x_1$ der Länge n auf dem sonst leeren Band steht mit Kopf von T auf der letzten Ziffer von w'. Begründen Sie knapp Ihre Konstruktionsidee.

2. Geben Sie ein Verfahren an, das zu jeder deterministischen Turingmaschine $T = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ eine eingeschränkt speicherfähige Turingmaschine $T' = (Q', \Sigma, \Gamma, \delta', q_0, \square, F) \in E$ liefert, so dass für die akzeptierten Sprachen $L(T) = L(T')$ gilt.

Lösung

1. Wir definieren zu jedem Element $x \in \Sigma$ einen (indizierten) Zustand q_x mit $q_x \neq q_y$ falls $x \neq y$.

   Seien $Q = \{q_0, f\} \cup \{q_x \,;\, x \in \Sigma\}$ mit $|Q| = |\Sigma| + 20$. Sei $F = \{f\}$.
   Wir definieren die Übergangsfunktion wie folgt:

$$\begin{array}{rclcrcll} \delta(q_0, \square) & = & (f, \square, L)\,, & & \delta(q_0, x) & = & (q_x, x, R)\,, & \forall x \in \Sigma \\ \delta(q_x, \square) & = & (f, \square, L)\,, & & \delta(q_x, y) & = & (q_0, x, N)\,, & \forall x, y \in \Sigma \end{array} \qquad (2\text{P})$$

2. Sei $Q' = Q \cup \{(q, x) \,;\, q \in Q, x \in \Gamma\}$. Wir schreiben q_x für (q, x) und definieren δ' wie folgt:

   Seien $q \in Q \setminus F$, $p \in Q$, $x, y \in \Gamma$, $d \in \{L, R, N\}$
   und es gelte $\delta(q, x) = (p, y, d)$. Dann und nur dann definieren wir δ', und zwar

$$\delta'(q, x) = (p_x, y, N) \quad \text{und} \quad \delta'(p_x, y) = (p, y, d)\,. \qquad (2\text{P})$$

Hausaufgabe 4 (4 Punkte)

Sei $\Sigma = \{*, \#\}$. Wir kodieren natürliche Zahlen $n \in \mathbb{N}$ als Folge $**\ldots*$ der Länge n, d. h. $|**\ldots*| = n$, und stellen Paare $x, y \in \{*\}^*$ als Wort $x\#y \in \Sigma^*$ dar.
Wir betrachten für $x, y, z \in \{*\}^*$ die modifizierte Subtraktion $|z| = |x| \dot{-} |y|$.
(Man beachte $1 \dot{-} 2 = 0$.)

1. Definieren Sie durch Angabe der Übergangsfunktion δ eine deterministische Turingmaschine $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$, die für $x, y, z \in \{*\}^*$ die modifizierte Subtraktion $|z| = |x| \dot{-} |y|$ wie folgt durchführt:

   Startkonfiguration: $(\epsilon, q_0, x\#y)$.
   Endkonfiguration: $(\square^k z, q_e, \#r\square^l)$, mit $q_e \in F$, $r \in \{*\}^*$ und $k, l \in \mathbb{N}$.

   Es gilt: $(\epsilon, q_0, x\#y) \rightarrow_M^* (\square^k z, q_e, \#r\square^l)$.

   Beschreiben Sie kurz die Konstruktionsidee für Ihre Maschine.

2. Gilt für Ihre Maschine M bei beliebiger Eingabe $w \in \Sigma^*$ die Gleichung $L(M) = \{x\#y \,;\, x, y \in \{*\}^*\}$? Begründen Sie Ihre Antwort!

Lösung

1.
$$\begin{aligned}
\delta(q_0,*) &= (q_0,*,R)\,, & \delta(q_0,\#) &= (s,\#,R)\,,\\
\delta(s,*) &= (s,*,R)\,, & \delta(s,\square) &= (t,\square,L)\,,\\
\delta(t,*) &= (l,\square,L)\,, & \delta(t,\#) &= (q_e,\#,N)\,,\\
\delta(l,*) &= (l,*,L)\,, & \delta(l,\#) &= (l,\#,L)\,,\\
\delta(l,\square) &= (q_1,\square,R)\,,\\
\delta(q_1,*) &= (q_0,\square,R)\,, & \delta(q_1,\#) &= (q_e,\#,N)\,.
\end{aligned}$$

Konstruktionsidee: Die äußeren Zeichen werden paarweise gelöscht bis entweder x oder y erstmalig vollständig gelöscht sind. (2P)

2. Ja! Begründung:

Offenbar gilt zunächst $\{x\#y\,;\, x,y \in \{*\}^*\} \subseteq L(M)$, da M für jede korrekte Eingabe hält.

Falls die Eingabe w kein Zeichen # enthält, dann überschreitet M beim ersten Lauf nach rechts im Zustand q_0 oder r das rechte Ende von w und trifft dort auf $\square$. Dann hält M ohne in den Endzustand zu gehen, weil kein Übergang definiert ist.

Falls w zwei oder mehr Zeichen # enthält, dann trifft M im ersten Lauf nach rechts im Zustand s auf das Zeichen #. Der Übergang ist nicht definiert und M hält ebenfalls nicht im Endzustand. (2P)

Hausaufgabe 5 (4 Punkte)

Zeigen Sie durch Konstruktion geeigneter linear beschränkter Automaten, dass die Klasse der kontextsensitiven Sprachen abgeschlossen ist gegenüber Durchschnittsbildung.

Lösung

Wir wollen voraussetzen, dass eine Turingmaschine $T = (Q,\Sigma,\Gamma,\delta,q,\square,F)$ mit Eingabealphabet Σ und Bandalphabet Γ für $\Sigma \subseteq \Sigma'$ und $\Gamma \subseteq \Gamma'$ in eine Turingmaschine $T' = (Q,\Sigma',\Gamma',\delta',q,\square,F)$ eingebettet werden kann mit gleicher akzeptierter Sprache $L(T') = L(T)$, Disjunktheit von Σ' mit Q und $\{\square\}$ sei dabei vorausgesetzt.

Seien $L_1 \subseteq \Sigma_1^*, L_2 \subseteq \Sigma_2^*$ kontextsensitive Sprachen, die von den LBA's
$A_1 = (Q_1,\Sigma_1,\Gamma_1,\delta_1,q_1,\square,F_1)$ bzw. $A_2 = (Q_2,\Sigma_2,\Gamma_2,\delta_2,q_2,\square,F_2)$ akzeptiert werden. Aufgrund der Bemerkung eingangs dürfen wir die Gleichheit der Alphabete Σ_1 und Σ_2 für die Sprachen und Automaten annehmen. Wir setzen also $\Sigma = \Sigma_1 = \Sigma_2$. Im Folgenden konstruieren wir einen LBA $A' = (Q',\Sigma,\Gamma',\delta',q',F')$ mit $L(A') = L_1 \cap L_2$.

Die Idee für A' ist, zunächst den Automaten A_1 auf der Eingabe rechnen zu lassen. Falls dann A_1 einen Endzustand erreicht, führt man A_2 auf der Eingabe aus und akzeptiert, wenn A_2 ebenfalls einen Endzustand erreicht. Allerdings überschreibt ja A_1 i.A. das Band. Wir müssen also die Eingabe zusätzlich auf dem Band speichern, üblicherweise, indem wir das Bandalphabet mit dem Produkt $\Gamma_1 \times \Gamma_2$ erweitern und beispielsweise in der zweiten Komponente der neuen Symbole die ursprüngliche Eingabe speichern. A_1 arbeitet dann ausschließlich auf der ersten Komponente. Wenn A_1 im Endzustand endet, dann muss der Kopf zurücklaufen auf das erste Eingabezeichen, um A_2 zu starten.

(4P)

Zusatzaufgabe 8 (Für Interessierte)

Es wird erzählt, dass in einem kleinen Dorf in Bayern ein alter Barbier lebt, der alle diejenigen Männer im Dorf rasiert, die sich nicht selbst rasieren.
Warum ist die Frage nach dem Alter des Barbiers sinnlos?
Formalisieren Sie den Sachverhalt und weisen Sie nach, dass die Erzählung eine Lüge enthält!

Lösung

Informelle Präsentation von Sachverhalten hat oft den scheinbaren Vorteil leichter Verständlichkeit, macht aber meist Hinweise zum „richtigen“ Verständis notwendig. Beispielsweise müssten wir klarstellen, dass der Barbier hier männlich zu sein hat. Man muss auch klarstellen, dass der Barbier diejenigen nicht rasiert, die sich selbst rasieren.

Die Frage nach dem Alter des Barbiers ist sinnlos, weil der Barbier nicht existiert. Dies können wir sagen, ohne den Erzähler zu kennen und ohne darüber mutmaßen zu müssen, ob es nicht einen Erzähler geben könnte, der vielleicht tatsächlich einen solchen Barbier kennt. Denn wir können logisch herleiten, dass der Barbier nicht existiert. Und das hat nichts mit einem Paradoxon zu tun. Es ist eine schlichte, logische Tatsache, die wir wie folgt beweisen.

Wir nehmen eine nichtleere Menge D an. Es sei $R \subseteq D \times D$ eine Relation, und wir sagen „x rasiert y“ für $(x, y) \in R$.
Die Menge A aller „Männer, die sich selbst rasieren“ bzw. B aller „Männer, die sich nicht selbst rasieren“ ist dann gegeben durch

$$A = \{x \in D\,;\, (x, x) \in R\} \quad \text{bzw.} \quad B = \{x \in D\,;\, (x, x) \notin R\}\,.$$

Die Formalisierung des Sachverhalts und die Behauptung der enthaltenen Lüge ist gegeben durch die folgende prädikatenlogische Aussage:

$$\neg\,(\exists b \in D\ \forall y \in A\, \forall z \in B)[(b, y) \notin R \wedge (b, z) \in R]\,.$$

Wir zeigen diese Aussage mit Widerspruchsbeweis und nehmen ein $b \in D$ an, so dass gilt $\forall y \in A\, \forall z \in B)[(b, y) \notin R \wedge (b, z) \in R]$. Da $A \cup B = D$, gilt einer der beiden Fälle $b \in A$ oder $b \in B$.
Fall 1, $b \in A$:
Als Mann des Dorfes rasiert sich b selbst. Dann aber darf er sich als Barbier nicht rasieren, $(b, b) \notin R$, d. h. $b \notin A$. Widerspruch!
Fall 2, $b \in B$:
Als Mann des Dorfes rasiert sich b nicht selbst. Dann aber muss er sich als Barbier rasieren, $(b, b) \in R$, d. h. $b \notin B$. Widerspruch!

Hinweis: *Die Vorbereitungsaufgaben bereiten die Tutoraufgaben vor und werden in der Zentralübung unterstützt. Tutoraufgaben werden in den Übungsgruppen bearbeitet. Hausaufgaben sollen selbstständig bearbeitet und zur Korrektur und Bewertung abgegeben werden.*

Vorbereitung 1

1. Beweisen oder widerlegen Sie: Wenn A semi-entscheidbar und B entscheidbar ist, dann ist $A \setminus B$ semi-entscheidbar.

2. Gegeben seien zwei entscheidbare Prädikate $P(x, y)$ und $Q(x, y)$. Zeigen Sie, dass $R(x, y) = P(x, y) \wedge Q(x, y)$ entscheidbar ist.

3. Ist jede Teilmenge einer rekursiven Sprache rekursiv aufzählbar? Beweis!

Lösung

1. Die Aussage ist wahr, und wir geben einen Algorithmus an, der $\chi'_{(A\setminus B)}$ berechnet:

   > Gegeben ein w, entscheide zunächst, ob $w \in B$. Falls ja, gehe in eine Endlosschleife. Falls nicht, berechne χ'_A und gib (bei Terminierung) das Ergebnis zurück.

   So berechnet man offensichtlich die Funktion

   $$\begin{aligned} f(w) &= \begin{cases} \bot & \text{falls } \chi'_B(w) = 1 \\ \chi'_A(w) & \text{sonst} \end{cases} \\ &= \begin{cases} 1 & \text{falls } w \notin B \text{ und } w \in A \\ \bot & \text{sonst} \end{cases} \\ &= \chi'_{(A\setminus B)}(w)\,. \end{aligned}$$

2. Sei ein nichtleeres Alphabet Σ gegeben, und seien P, Q zweistellige Prädikate über Σ^* mit den berechenbaren charakteristischen Funktionen χ_P bzw. χ_Q der Funktionalität $\Sigma^* \times \Sigma^* \to \{0, 1\}$. Dann lautet die charakteristische Funktion χ_R für $R(x, y) = P(x, y) \wedge Q(x, y)$ wie folgt.

   $$\chi_R(x, y) = \chi_P(x, y) \cdot \chi_Q(x, y)\,.$$

   Offensichtlich ist χ_R berechenbar.

3. Es existiert eine Sprache $L \subseteq \Sigma^*$, die nicht rekursiv aufzählbar ist, z.B. L_d. Da aber Σ^* rekursiv ist, kann nicht jede Teilmenge von Σ^* rekursiv aufzählbar sein. L ist ein Gegenbeispiel.

Vorbereitung 2

Zeigen Sie, dass man die folgende Anweisung durch ein LOOP-Programm simulieren kann, das kein IF-Konstrukt enthält: **IF** $x_i \leq x_j$ **THEN** P_1 **ELSE** P_2 **END**.

Lösung

Das folgende LOOP - Programm stützt sich auf das LOOP-Programm IF $x = 0$ THEN P END, das in der Vorlesung durch ein LOOP-Programm definiert wird. Seien $x_1, x_2, \notin \{x_i, x_j\}$.

$x_1 := x_i$; LOOP x_j DO $x_1 := x_1 - 1$; //Berechnung von $x_1 = x_i \dot{-} x_j$
$x_2 := 0$;
IF $x_1 = 0$ THEN P_1 ; $x_2 := 1$ END ;
IF $x_2 = 0$ THEN P_2 END

Vorbereitung 3

Zeigen Sie durch Rückführung auf die Definition, dass die folgenden Funktionen primitiv-rekursiv sind:

$$iszero(x) = \begin{cases} 1 & \text{falls } x = 0 \\ 0 & \text{sonst} \end{cases}, \qquad eq(x, y) = \begin{cases} 1 & \text{falls } x = y \\ 0 & \text{sonst} \end{cases}.$$

Lösung

Wir schreiben die primitiv-rekursiven Basis-Projektionsfunktionen $proj_{k,i}(x_1, x_2, \ldots, x_k)$ als $\pi_i^k(x_1, x_2, \ldots, x_k)$.

s sei die Basis-Nachfolgerfunktion.

Wir definieren zunächst die primitiv-rekursiven arithmetischen Funktionen $add(x, y)$, $mult(x, y)$, $\dot{-}(x, y)$ bzw. in üblicher Infixschreibweise $x + y$, $x \cdot y$, $x \dot{-} y$

und Semantik

$$\dot{-}(x, y) \;=\; \max\{x - y, 0\}\,.$$

Addition:

Lesbare Kurzschrift zuerst:

$$\begin{aligned} 0 + y &= y\,, \\ (x + 1) + y &= s(x + y)\,. \end{aligned}$$

oder

$$\begin{aligned} add(0, y) &= y\,, \\ add(x + 1, y) &= s(add(x, y)) \end{aligned}$$

Syntaktisches Format:

$$\begin{aligned} h(r, x, y) &= s(\pi_1^3(r, x, y))\,, \\ add(0, y) &= \pi_1^1(y)\,, \\ add(x + 1, y) &= h(add(x, y), x, y)\,. \end{aligned}$$

Multiplikation:

Lesbare Kurzschrift zuerst:

$$\begin{aligned} 0 \cdot y) &= 0\,, \\ (x+1) \cdot y &= (x \cdot y) + y\,. \end{aligned}$$

oder

$$\begin{aligned} mult(0,y) &= 0\,, \\ mult(x+1,y) &= add(mult(x,y),y) \end{aligned}$$

Syntaktisches Format:

$$\begin{aligned} k(y) &= 0\,, \\ h(r,x,y) &= add(\pi_1^3(r,x,y),\pi_3^3(r,x,y))\,, \\ mult(0,y) &= k(y)\,, \\ mult(x+1,y) &= h(mult(x,y),x,y)\,. \end{aligned}$$

Modifizierte Subtraktion:

Lesbare Kurzschrift zuerst:

$$\begin{aligned} pred(0) &= 0\,, \\ pred(x+1) &= x\,, \\ x \dot{-} 0 &= x\,, \\ x \dot{-} (y+1) &= pred(x \dot{-} y)\,. \end{aligned}$$

Syntaktisches Format:

$$\begin{aligned} h_1(r,x) &= \pi_2^2(r,x)\,, \\ pred(0) &= 0\,, \\ pred(x+1) &= h_1(pred(x),x)\,, \\ h_2(r,x,y) &= pred(\pi_1^3(r,x,y))\,, \\ \dot{-}^R(0,x) &= \pi_1^1(x)\,, \\ \dot{-}^R(y+1,x) &= h_2(\dot{-}^R(y,x),y,x)\,, \\ \dot{-}\,(x,y) &= \dot{-}^R(y,x)\,. \end{aligned}$$

Lösung in lesbarer Form:

$$\begin{aligned} iszero(0) &= 1\,, \\ iszero(x+1) &= 0\,, \\ eq(x,y) &= iszero((x \dot{-} y) + (y \dot{-} x))\,. \end{aligned}$$

Mit welchen Regeln kann man stets eine lesbare Form erreichen?

Dieser Frage werden wir in VA 5 nachgehen.

Vorbereitung 4

Sei $f(x, y)$ primitiv rekursiv. Zeigen Sie mit Hilfe der Projektionsfunktionen $\pi_i^k := proj_{k,i}$ zusammen mit der (nicht erweiterten) Komposition, dass die Funktion g mit $g(x, y) = f(y, x)$ für alle $x, y \in \mathbb{N}$ ebenfalls primitiv-rekursiv ist.

Lösung

$g(x, y) = f(\pi_2^2(x, y), \pi_1^2(x, y))$.

Vorbereitung 5

Wir bezeichnen f als eine erweiterte Komposition der Funktionen $g_1, \ldots, g_k$, falls $f(x_1, \ldots, x_n) = t$, so dass t ein Ausdruck ist, der nur aus den Funktionen $g_1, \ldots, g_k$ und den Variablen $x_1, \ldots, x_n$ besteht.

Sei t_0 ein funktionaler Ausdruck, der nur primitiv-rekursive Funktionen und Variable x_i enthält. t enthalte nur $f(m, \overline{x})$ mit einem Variablenvektor $\overline{x}$, primitiv-rekursive Funktionen, m und Variable x_i. Dann heißen die Gleichungen

$$f(0, \overline{x}) = t_0, \;\; f(m + 1, \overline{x}) = t$$

das erweiterte Schema der primitiven Rekursion.

Man zeige:

1. Eine erweiterte Komposition von primitiv-rekursiven Funktionen ist wieder primitiv-rekursiv.
2. Das erweiterte Schema der primitiven Rekursion führt nicht aus der Menge der primitiv-rekursiven Rekursionen heraus.

Lösung

Man beachte:

$f(f(m, \overline{x}), \overline{x})$ ist im erweiterten Schema nicht zulässig.

Bemerkung:
Auch eine Erweiterung auf Systeme von Gleichungen ist möglich, in denen dann auch für ein konkretes k die Teilausdrücke $f(m - 1, \overline{x}), \ldots, f(m - k, \overline{x})$ vorkommen dürfen.
(Siehe Tutoraufgaben, Kodierung mit Paarfunktionen)

Die zweite Behauptung zeigt man mit erweiterter Komposition.

Die erste Behauptung zeigt man mit Induktion über den Aufbau des funktionalen Ausdrucks t.

Beispiel:

$f(x, y) = g_1(x, g_2(y, g_3(x)))$.

Umsetzung in syntaktisches Format:

$$\begin{aligned} h_1(x, y) &= g_3(\pi_1^2(x, y)), \\ h_2(x, y) &= g_2(\pi_2^2(x, y), h_1(x, y)), \\ f(x, y) &= g_1(\pi_1^2(x, y), h_2(x, y)). \end{aligned}$$

Tutoraufgabe 1

Zeigen Sie, dass man die folgenden Anweisungen durch WHILE-Programme, wie sie in der Vorlesung definiert wurden, simulieren kann:

1. $x_i := x_j + x_k$ (Addition zweier Variablen),
2. $x_i := x_j \dot{-} x_k$ (Bedingte Subtraktion, d. h. $x_j \dot{-} x_k = 0$ für $x_j \leq x_k$).

Lösung

Wir gehen davon aus, dass wir neue Variablen einführen können. Wir markieren solche Variablen durch einen Querbalken. Wir wollen auch selbsterklärende Schreibvarianten zulassen.

1. Folgendes WHILE-Programm implementiert die Addition $x_i := x_j + x_k$:

$\bar{x}_1 := x_j + 0$;
$\bar{x}_2 := x_k + 0$;
while $\bar{x}_2 \neq 0$ **do**
  $\bar{x}_1 := \bar{x}_1 + 1$;
  $\bar{x}_2 := \bar{x}_2 \dot{-} 1$
end;
$x_i := \bar{x}_1 + 0$

2. Folgendes WHILE-Programm implementiert die Subtraktion $x_i := x_j \dot{-} x_k$:

$\bar{x}_1 := x_j + 0$;
$\bar{x}_2 := x_k + 0$;
while $\bar{x}_2 \neq 0$ **do**
  $\bar{x}_1 := \bar{x}_1 \dot{-} 1$;
  $\bar{x}_2 := \bar{x}_2 \dot{-} 1$
end;
$x_i := \bar{x}_1 + 0$

Tutoraufgabe 2

Zeigen Sie durch Rückführung auf die Definition, dass die folgenden Funktionen primitiv-rekursiv sind.

1. $\mathit{tower}(n) = 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}$ (d.h. $2^{(2^{(2^{\cdot^{\cdot^{\cdot^{2}}}})})}$, Turm der Höhe n),
2. $\mathit{ifthen}(n, a, b)$ mit

$$\mathit{ifthen}(n, a, b) = \begin{cases} a & n \neq 0\,, \\ b & n = 0\,. \end{cases}$$

Lösung

Die Konstante 1 und die Multiplikation $mult(x, y)$ ist nach VA 3 PR. Wir benutzen außerdem die erweiterte Komposition von primitiv rekursiven Funktionen

1. $$\begin{aligned} twopow(0) &= 1 \\ twopow(n+1) &= mult(twopow(n), 2). \\ tower(0) &= 1 \\ tower(n+1) &= twopow(tower(n)). \end{aligned}$$

2. $$\begin{aligned} ifthen(0, a, b) &= b \\ ifthen(n+1, a, b) &= a. \end{aligned}$$

Tutoraufgabe 3

Sei $f : \mathbb{N}_0 \to \mathbb{N}_0$ diejenige Funktion, die für alle $n \in \mathbb{N}_0, n \neq 0$ durch die Rekursion

$$f(n+1) = f(n) \cdot f(n-1)$$

mit den Startwerten $f(0) = 1$ und $f(1) = 2$ definiert ist.

1. Zeigen Sie, dass f primitiv-rekursiv ist, indem Sie ein LOOP-Programm angeben.
2. Zeigen Sie durch Rückführung auf die Definition, dass f primitiv-rekursiv ist.

Lösung

1. Das folgende LOOP-Programm basiert auf den Variablen $x_0, \ldots, x_5$. In x_0 wird das Ergebnis $f(n)$ ausgegeben, x_1 enthält beim Start das Argument n.

   $x_2 := x_1 - 1;$
   $x_3 := 1;\ x_4 := 2;$
   $LOOP\ x_2\ DO$
   $\quad x_5 := x_4 * x_3;$
   $\quad x_3 := x_4;\ x_4 := x_5;$
   $END;$
   $x_0 := x_5$
   $IF\ x_1 = 0\ THEN\ x_0 := 1\ END;$
   $IF\ x_1 = 1\ THEN\ x_0 := 2\ END$

   Wenn wir Satz der Vorlesung benutzen, dann folgt aus der Existenz eines LOOP-Programms für die Funktion f die primitive Rekursivität von f.

2. Wir stützen uns auf das erweiterte Schema zur Definition primitiv-rekursiver Funktionen.

   Die Schwierigkeit der Anwendung der Schemata der primitiven Rekursion auf die Funktion f besteht darin, dass die Funktionswerte $f(n+1)$ nicht allein von $f(n)$ sondern auch von $f(n-1)$ abhängen. Dieses Problem kann man angehen, in dem man Tupel von natürlichen Zahlen als natürliche Zahl kodiert.

In Tupel- bzw. Vektorschreibweise liest sich die Rekursion wie folgt. Wir suchen eine Funktion f, so dass $(f(0), f(1))^T = (1, 2)^T$ und für alle $n \in \mathbb{N}_0$ die folgende Gleichung gilt.

$$\begin{pmatrix} f(n+1) \\ f(n+2) \end{pmatrix} = \begin{pmatrix} f(n+1) \\ f(n+1) \cdot f(n) \end{pmatrix}$$

Wir kodieren die Tupel mit der bijektiven Paarfunktion $p : \mathbb{N}_0^2 \to \mathbb{N}_0$ der Vorlesung und benützen die Projektionen c_1 und c_2 der Umkehrfunktion von p. Man beachte, dass sowohl p als auch die Projektionen c_1 und c_2 primitiv-rekursiv sind.

Wir betrachten die Funktion $g : \mathbb{N}_0 \to \mathbb{N}_0$ mit

$$g(n) = p(f(n), f(n+1))$$

und zeigen deren primitive Rekursivität mit erweiterten Rekursionsschemata wie folgt.

$$\begin{aligned} g(0) &= p(1, 2)\,, \\ g(n+1) &= p(\, c_2(g(n))\,,\, c_2(g(n)) \cdot c_1(g(n))\,)\,. \end{aligned}$$

Nun gilt aber auch

$$f(n) = c_1(g(n))\,.$$

Damit ist die primitive Rekursivität von f gezeigt.

Tutoraufgabe 4

Wir wollen untersuchen, ob sich (ähnlich wie bei den WHILE-Programmen) jedes LOOP-Programm in eine „Normalform“

$$\texttt{LOOP}\ X\ \texttt{DO}\ P\ \texttt{END}$$

bringen lässt, so dass P keine Schleifen mehr enthält.
Wir bezeichnen mit $V(P)$ die Menge der Variablen, die in P vorkommen (diese Menge ist stets endlich). Für einen gegebenen Programmlauf bezeichnen wir mit $[x]$ den Wert der Variablen x beim Start des Programms, und mit $[x]'$ den Wert der Variablen nach Programmende.

1. Zeigen Sie: Für jedes LOOP-Programm P ohne Schleifen gibt es eine Konstante k, so dass gilt:
$$\max_{x \in V(P)} [x]' \leq \max_{x \in V(P)} [x] + k\,.$$

2. Zeigen Sie, dass es kein LOOP-Programm in Normalform geben kann, welches die Quadratfunktion $n \mapsto n^2$ berechnet.

Lösung

1. Die Beziehung zeigen wir mit Induktion über den Aufbau von P. Da wir annehmen, dass P keine Schleifen hat, müssen wir den LOOP-Fall nicht betrachten:

   - Für $P = \ x_i := x_j + c$ gilt offensichtlich $[x_i]' \leq [x_j] + c$ und, da alle anderen Variablen ihren Wert behalten, gilt auch $\max_{x \in V(P)}[x]' \leq \max_{x \in V(P)}[x] + c$. Wir wählen also $k = c$.
   - Für $P = \ x_i := x_j - c$ können wir $k = 0$ wählen, da keine Variable erhöht wird, und somit auch das Maximum nicht wachsen kann.
   - Für $P = \ P_1; P_2$ können wir per Induktionshypothese Konstanten k_1, k_2 annehmen, so dass die Variableninhalte im ersten Teilprogramm maximal um k_1 wachsen, und im zweiten Teil maximal um k_2. Somit kann man $k = k_1 + k_2$ setzen.

2. Wir nehmen an, Q sei ein LOOP-Programm in Normalform, welches die Quadratfunktion berechnet. Dann ist

   $$Q = \texttt{LOOP}\ x_0\ \texttt{DO}\ P\ \texttt{END}$$

   und P enthält keine Schleife. Wir können hierbei annehmen, dass die einzige Eingabevariable x_0 gleichzeitig die Schleifenvariable ist, denn ansonsten würde die Schleife gar nicht ausgeführt werden (da alle anderen Variablen anfangs auf Null gesetzt sind), und Q könnte sicher nicht quadrieren.

   Nach Aufgabenteil 1 gibt es ein k, so dass nach jeder Ausführung von P gilt, dass $\max_{v \in V}[v]' \leq \max_{v \in V}[v] + k$. Da P insgesamt $[x_0]$ mal durchlaufen wird, gilt für das Gesamtprogramm $\max_{v \in V}[v]' \leq \max_{v \in V}[v] + [x_0]k$. Da bei Programmbeginn alle Variablen außer x_0 auf Null gesetzt sind, haben wir $\max_{v \in V}[v] = [x_0]$ und somit $\max_{v \in V}[v]' \leq [x_0](k+1)$.

   Ruft man nun Q mit der Eingabe $k + 2$ auf, dann gilt:

   $$[x_0]' \leq \max_{v \in V}[v]' \leq (k+2)(k+1) < (k+2)^2$$

   Somit hat Q die Eingabe nicht quadriert, im Widerspruch zur Annahme.