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

Sommersemester 2015
Lösungsblatt 12
17. Juli 2015

Theoretische Informatik

Hausaufgabe 1 (4 Punkte)

Der Binomialkoeffizient $binom(n,m) = \binom{n}{m}$ ist eine Funktion von $\mathbb{N}_0^2$ in $\mathbb{N}_0$ mit den Eigenschaften $\binom{n}{0} = 1$, $\binom{0}{m} = 0$ für $n, m \in \mathbb{N}_0$ mit $m > 0$. Der Binomialkoeffizient erfüllt für alle $n, m \in \mathbb{N}_0$ die Rekursionsgleichung

$$\binom{n+1}{m+1} = \binom{n}{m+1} + \binom{n}{m}.$$

Für jede natürliche Zahl m_0 betrachten wir die Funktion $b_{m_0} : \mathbb{N}_0 \to \mathbb{N}_0$ mit $b_{m_0}(n) = binom(n, m_0)$. Zeigen Sie durch Induktion über m_0, dass alle Funktionen $b_{m_0}(n)$ mit $m_0 \in \mathbb{N}_0$ primitiv-rekursiv (als Funktion in n) sind.

Lösung

$m_0 = 0$: $b_{m_0}(n) = \binom{n}{0} = 1$ ist offenbar primitiv rekursiv.
Sei $b_{m_0}(n)$ primitiv rekursiv für ein $m_0 \geq 0$. Wir zeigen, dass $b_{m_0+1}(n)$ primitiv rekursiv ist, wie folgt:
Nach Voraussetzung ist $b_{m_0}(n)$ PR und bekanntlich ist die Addition PR.
Erweitertes Rekursionsschema für $b_{m_0+1}(n)$:

$$\begin{aligned} b_{m_0+1}(0) &= 0\,, \\ b_{m_0+1}(n+1) &= b_{m_0+1}(n) + b_{m_0}(n)\,. \end{aligned}$$

(4P)

Hausaufgabe 2 (4 Punkte)

Sei $\Sigma = \{0, 1\}$. $M_w[x]\downarrow$ bedeutet, dass die durch $w \in \{0,1\}^*$ kodierte Turingmaschine M_w bei Eingabe x hält, d.h. terminiert.

1. Wir betrachten das spezielle Halteproblem $K = \{w \in \Sigma^*\,;\, M_w[w]\downarrow\}$ und das Halteproblem auf leerem Band $H_0 = \{w \in \Sigma^*\,;\, M_w[\epsilon]\downarrow\}$.

   Zeigen Sie durch hinreichend genaue Spezifikation und Begründung einer Reduktionsabbildung (wie in den entsprechenden Beweisen der Vorlesung), dass H_0 reduzierbar ist auf K, d.h. $H_0 \leq K$.

2. Zeigen Sie, dass die Menge $R = \{w \in \Sigma^*\,;\, \varphi_w(0) = \perp\}$ unentscheidbar ist.
   Dabei sei φ_w diejenige (partielle) Funktion $\varphi_w : \mathbb{N}_0 \to \mathbb{N}_0$, die von der Turingmaschine M_w berechnet wird.

Lösung

1. Jedem w wird $f(w)$ zugeordnet, so dass die Maschine $M_{f(w)}$ jede Eingabe u löscht und dann auf das leere Band die Maschine M_w anwendet.

   $M_{f(w)}$ hält genau dann auf $f(w)$, wenn M_w auf dem leeren Band hält (2P)

2. Satz von Rice mit $F = \{\varphi_w\,;\, \varphi_w(0) = \perp\}$.

   Es gilt $F \neq \emptyset$ und die berechenbare Konstante 1 liegt nicht in F. (2P)

Hausaufgabe 3 (4 Punkte)

Sei Σ ein Alphabet und seien $A, B \subseteq \Sigma^*$ rekursiv auflistbare Sprachen. Zeigen Sie:

1. $L_1 := ABA$ ist rekursiv auflistbar.
2. $L_2 := A \cap B$ ist rekursiv auflistbar.

Hinweis: Die Cantorsche Paarfunktion bzw. die dazugehörigen Projektionen c_1 und c_2 könnten hilfreich sein.

Lösung

1. Wir zeigen zunächst, dass $C = AB$ auflistbar ist.

   Der Fall $C = \varnothing$ ist trivial. Wir betrachten den Fall $C \neq \varnothing$. Seien also f_1 und f_2 Funktionen, die A bzw. B aufzählen. Dann ist die Funktion $f(n) = f_1(c_1(n))f_2(c_2(n))$ total und berechenbar (denn f_1, f_2, c_1 und c_2 sind total und berechenbar) und zählt C auf:

   Da $f_1(c_1(n)) \in A$ und $f_2(c_2(n)) \in B$ für alle $n \in \mathbb{N}_0$, ist $f(n) \in C$ für alle $n \in \mathbb{N}_0$. Umgekehrt gilt für ein $x \in C$, dass $y \in A$ und $z \in B$ existieren mit $x = yz$. Da f_1 und f_2 die Sprachen A und B aufzählen, gibt es $k, l \in \mathbb{N}_0$ mit $f_1(k) = y$ und $f_2(l) = z$. Da die Cantor'sche Paarfunktion bijektiv ist, gibt es ein $n \in \mathbb{N}_0$, so dass $c_1(n) = k$ und $c_2(n) = l$ gibt; f zählt also auch alle Elemente von C auf.

   Nun kann man analog die Aufzählbarkeit von $L_1 = CA$ beweisen. (2P)

2. Ist $L_2 = \varnothing$, so ist es rekursiv auflistbar nach Definition. Ist $L_2 = A \cap B \neq \varnothing$, so gilt $A \neq \varnothing$ und $B \neq \varnothing$ und es gibt ein $d \in L_2$ und Funktionen f_1 und f_2, die A und B aufzählen. Wir definieren nun f wie folgt.

   $$f(n) = \begin{cases} f_1(c_1(n)) & \text{wenn } f_1(c_1(n)) = f_2(c_2(n)) \\ d & \text{sonst} \end{cases}$$

   Die Funkion ist total und berechenbar (denn f_1 und f_2 sind ebenfalls total und berechenbar) und zählt L_2 auf: Gilt $f_1(k) = f_2(l)$ für $k, l \in \mathbb{N}_0$, so ist $f_1(k)$ in A und B, also auch in ihrem Schnitt enthalten. Weiterhin gilt auch $d \in L_2$. Alle von f aufgezählten Werte sind also in L_2 enthalten.

   Da f_1 die Menge L_1 und f_2 die Menge B aufzählt, existieren umgekehrt für jedes $x \in L_2$ Zahlen $k, l \in \mathbb{N}_0$, so dass $f_1(k) = f_2(l) = x$ gilt. Da die Cantor'sche Paarfunktion bijektiv ist, gibt es ein $n \in \mathbb{N}_0$, so dass $c_1(n) = k$ und $c_2(n) = l$ gibt; f zählt also auch alle Elemente von L_2 auf. (2P)

Hausaufgabe 4 (4 Punkte)

Sei $\Sigma = \{0,1\}$. Geben Sie jeweils ein Beispiel für die folgenden Objekte an. Falls kein solches Objekt existiert, begründen Sie dies.

1. Eine Funktion $f : \mathbb{N}_0 \to \mathbb{N}_0$, die primitiv-rekursiv ist, aber deren Definitionsbereich (also $\{n \in \mathbb{N}_0\,;\, f(n) \neq \perp\}$) endlich ist.
2. Eine Funktion $f : \Sigma^* \to \Sigma^*$, die total ist und für die $\{w \in \Sigma^*\,;\, \varphi_w = f\}$ entscheidbar ist.
3. Ein unentscheidbarer Wertebereich einer berechenbaren Funktion.

Lösung

1. Existiert nicht, da jede primitiv rekursive Funktion total ist und somit Definitionsbereich $\mathbb{N}_0$ hat. (1P)
2. Eine beliebige nicht-berechenbare und totale Funktion, denn dann ist die genannte Menge leer und somit entscheidbar. (1P)
3. Seien $c : \Sigma^* \to \mathbb{N}_0$ eine bijektive Kodierung von Σ^*, so dass sowohl c als auch die Umkehrfunktion c^{-1} primitiv-rekursiv ist (Gödelisierung), und $L \subset \Sigma^*$ eine semi-entscheidbare, aber nicht entscheidbare Sprache, wie z.B. das spezielle Halteproblem. Nun gibt es eine rekursive Aufzählung f von L.

   Wir definieren $g = c \circ f$. Dann ist der Wertebereich von g nicht entscheidbar. (2P)

Hausaufgabe 5 (4 Punkte)

Wahr oder falsch? Begründen Sie im Folgenden Ihre Antworten möglichst knapp!

1. Wenn f berechenbar ist, dann ist $A_f := \{w \in \Sigma^*\,;\, f(w) \neq \perp\}$ semi-entscheidbar.
2. Für das spezielle Halteproblem $K = \{w \in \{0,1\}^*\,;\, M_w[w]\downarrow\}$ und eine beliebige Sprache A gilt: Wenn $K \cap A$ entscheidbar ist, dann ist A endlich.
3. Für jede Turingmaschine M ist die Funktion

$$f_M(x) = \begin{cases} 1 & \text{falls } M \text{ auf allen Eingaben hält} \\ 0 & \text{sonst} \end{cases}$$

   berechenbar.
4. Wenn f und g primitiv-rekursiv sind, und $f(x) = g(h(x))$ für alle x gilt, dann ist auch h primitiv-rekursiv.

Lösung

(1 Punkt pro Teilaufgabe)

1. (w): $\chi'_{A_f}(x) = \begin{cases} 1 & \text{falls } f(x) \neq \perp \\ \perp & \text{sonst} \end{cases}$ ist berechenbar, da f berechenbar ist.

2. (f): Gegenbeispiel: $A = \overline{K}$. Dann ist $K \cap A = \emptyset$ und deshalb entscheidbar, obwohl A unendlich ist.

3. (w): Für jede Turingmaschine M ist f_M eine konstante Funktion und als solche natürlich berechenbar.

4. (f): Gegenbeispiel: $f(x) = 0$, $g(x) = 0$, und h beliebig und nicht PR.

Zusatzaufgabe 10 (wird nicht korrigiert)

Geben Sie für jede der folgenden Mengen an, ob sie entscheidbar ist oder nicht. Beweisen Sie ihre Behauptungen.

1. $L_1 = \{w \in \Sigma^* \,;\, \varphi_w(0) = 0\}$.

2. $L_2 = \{w \in \Sigma^* \,;\, \varphi_w(w) = w\}$.

3. $L_3 = \{w \in \Sigma^* \,;\, \varphi_0(0) = w\}$.

Lösung

1. Unentscheidbar nach Rice. Sei $F = \{f \,;\, f(0) = 0, f \text{ berechenbar}\}$. Dann ist die Identitätsfunktione $i \in F$, aber die überall undefinierte Funktion $\Omega \notin F$, und $L_1 = \{w \,;\, \phi_w \in F\}$. Damit sind die Bedinungen des Satzes erfüllt.

2. Unentscheidbar, aber der Satz von Rice ist nicht anwendbar. Reduktion z.B. von H_0: Für w, konstruiere TM, die $M_w[\epsilon]$ simuliert und bei Halten ihre ursprüngliche Eingabe zurückgibt. Andere Reduktion, die funktioniert: K.

3. Trivial entscheidbar, da einelementige Menge: $L_3 = \{\varphi_0(0)\}$ (eine geeignete Kodierung vorausgesetzt).

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

Warum kann man den Satz von Rice auf die folgende Menge nicht anwenden?

$$L = \{w \in \Sigma^* \,;\, \forall n \in \mathbb{N}_0 :\ \varphi_w(n) = \perp \text{ und } w \text{ ist ein Palindrom}\}\,.$$

Lösung

Hier ist der Satz von Rice nicht anwendbar, denn dafür müsste es eine Funktionenmenge F geben, so dass $L = \{w\,;\,\varphi_w \in F\}$. Es ist aber nicht ausgeschlossen, dass es Wörter v und w gibt, so dass $\varphi_v = \varphi_w$ und v ein Palindrom ist, aber w keines. Ob es solche Wörter tatsächlich gibt, hängt von der konkreten Codierung von Turingmaschinen in Wörter ab.

Vorbereitung 2

1. Wir betrachten das Postsche Korrespondenzproblem $P = (\,(1, c1), (abc, ab)\,)$.

   Bestimmen Sie alle Lösungen von P!

2. Sei $P = (p_1, p_2)$ ein Postsches Korrespondenzproblem über einem beliebigen Alphabet Σ mit $p_i = (x_i, y_i)$ und $|\,|x_i| - |y_i|\,| = 1$ für $i = 1, 2$.

   Zeigen Sie, dass P entscheidbar ist!

Lösung

1. Für die Menge L der Lösungen gilt $L = \{(i_1 i_2)^n\,;\, n \in \mathbb{N}_0, n \neq 0, i_1 = 2, i_2 = 1\}$.

2. Es gibt genau dann eine Lösung von P, wenn $1, 2$ oder $2, 1$ eine Lösung ist, d. h. wenn entweder $x_1x_2 = y_1y_2$ oder $x_2x_1 = y_2y_1$ gilt.

   Beweis:

   Wir nehmen an, dass $i_1 i_2 \ldots i_k$ eine Lösung von P ist. Zunächst gilt sicher $k \geq 2$, denn aus der Längenbedingung für die p_i folgt, dass weder p_1 noch p_2 selbst schon eine Lösung ist.

   Falls $i_1 \neq i_2$, dann folgt $|x_{i_1} x_{i_2}| = |y_{i_1} y_{i_2}|$. Mithin ist bereits $i_1 i_2$ eine Lösung, d. h. $1, 2$ oder $2, 1$ ist Lösung.

   Sei nun $i_1 = i_2$ und o. B. d. A. $i_1 = i_2 = 1$. Wir nehmen ebenfalls o. B. d. A. an, dass y_1 ein Präfix von x_1 ist, d. h. $x_1 = u_1 u_2 \ldots u_n u_{n+1}$ und $y_1 = u_1 u_2 \ldots u_n$ mit $u_1, \ldots, u_{n+1} \in \Sigma$. Da $i_1 i_2$ Teil einer Lösung ist, gilt

   $$\begin{aligned} x_1 x_1 &= y_1 y_1 u_n u_{n+1} \quad \text{und} \\ u_1 u_2 \ldots u_n u_{n+1} u_1 u_2 \ldots u_n u_{n+1} &= u_1 u_2 \ldots u_n u_1 u_2 \ldots u_n u_n u_{n+1} \end{aligned}$$

   Durch buchstabenweisen Vergleich folgt

   $$u_{n+1} = u_1 = u_2 = \ldots = u_{n-1} = u_n\,,$$

Wir erhalten $p_1 = (a^{n+1}, a^n)$ für irgendeinen Buchstaben $a \in \Sigma$.

In der Sequenz $i_1 i_2 \dots i_k$ muss es eine Teilsequenz $i_1 i_2 \dots i_l$ geben, die selbst Lösung ist und für die gilt $i_{l-1} = 2, i_l = 2$. Dann zeigt man analog $p_2 = (b^m, b^{m+1})$.

Da nun p_1 und p_2 irgendwo gematcht werden müssen, folgt $a = b$. Mithin folgt, dass $1, 2$ eine Lösung ist. Es folgt sogar, dass $2, 1$ ebenfalls eine Lösung ist.

Vorbereitung 3

Zeigen Sie, dass die polynomielle Reduzierbarkeit $\leq_p$ eine transitive Relation ist. Polynomielle Reduzierbarkeit bedeutet, dass die Reduktionsfunktion in polynomieller Zeit berechenbar ist.

Lösung

Seien $\Sigma_1, \Sigma_2, \Sigma_3$ Alphabete und $A \subseteq \Sigma_1^*, B \subseteq \Sigma_2^*, C \subseteq \Sigma_3^*$, so dass $A \leq_p B$ und $B \leq_p C$ gilt.
Wir zeigen, dass dann auch $A \leq_p C$ gilt, wie folgt.

Nach Definition der Reduzierbarkeit gibt es berechenbare Funktionen $f : \Sigma_1^* \to \Sigma_2^*$ und $g : \Sigma_2^* \to \Sigma_3^*$ (beide total in Σ_1^* bzw. Σ_2^*) mit entsprechenden, f bzw. g berechnenden DTM's F bzw. G, sowie Polynome p, q, so dass gilt

$$\begin{array}{rcl} \forall x \in \Sigma_1^*.\ \ \mathrm{TIME}_F(x) & \leq & p(|x|)\,, \\ \forall x \in \Sigma_2^*.\ \ \mathrm{TIME}_G(x) & \leq & q(|x|)\,. \end{array}$$

und außerdem noch $f^{-1}(B) = A$ und $g^{-1}(C) = B$ gilt.

Wir zeigen nun, dass die Funktionskomposition $g \circ f$ die Menge A polynomiell berechenbar auf C reduziert.

Zunächst gilt $(g \circ f)^{-1}(C) = f^{-1}(g^{-1}(C)) = f^{-1}(B) = A$, d. h. $x \in A \iff (g \circ f)(x) \in C$.

Dann gibt es eine Turingmaschine $K(F, G)$ die die Komposition von F und G im Wesentlichen als Simulation von F angewandt auf x und nachfolgend G angewandt auf die Ausgabe $f(x)$ ausführt.

Man kann die Schrittzahl der Simulation von F angewandt auf x mit $p(|x|)$ abschätzen.

Die Schrittzahl der Simulation von G angewandt auf $f(x)$ wird mit $q(|f(x)|)$ abgeschätzt. Da die Länge von $f(x)$ sicher höchstens gleich $|x| + p(|x|)$ ist, erhalten wir für die Abschätzung der Simulation von G angewandt auf $f(x)$ die Schranke $q(|x| + p(|x|))$.

Allerdings wird die Maschine K für die Hintereinanderschaltung von F und G noch Berechnungsschritte ausführen, die aber sicher so angelegt werden können, dass sie nur linear von $|x|$ und $|f(x)|$ abhängen. Eine zusätzliche Abhängikeit von $|(g \circ f)(x)|$ kann man vermeiden.

Zusammenfassend erhalten wir

$$\forall x \in \Sigma_1^*.\ \ \mathrm{TIME}_K(x) \ \leq \ p(|x|) + q(|x| + p(|x|)) + c(|x| + p(|x|))\,.$$

Offenbar ist $g \circ f$ polynomiell beschränkt.

Vorbereitung 4

Beantworten Sie kurz die folgenden Fragen:

1. Ist TIME_M für jede deterministische Turingmaschine M berechenbar?

2. PSPACE ist die Klasse all jener Probleme, die eine DTM mit „polynomiell viel Band in Abhängigkeit der Länge der Eingabe“ lösen kann. Gilt $\mathcal{P} \subseteq$ PSPACE?

Lösung

1. Nein, denn dann wäre das Halteproblem entscheidbar.

2. Ja, denn in polynomieller Zeit kann nur polynomiell viel Band beschrieben werden. Ob PSPACE $\subseteq \mathcal{P}$ gilt, ist ein offenes Problem.

Vorbereitung 5

Beweisen Sie:

1. $\mathcal{P}$ ist abgeschlossen unter Komplement.

2. Das Problem, zu entscheiden, ob ein gegebener Graph ein Dreieck enthält, ist in $\mathcal{P}$.

Lösung

1. Sei $A \in \Sigma^*$ in $\mathcal{P}$ und sei $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ eine Turingmaschine, die A in polynomieller Zeit entscheidet, d.h. die charakteristische Funktion χ_A berechnet. Dann ist $M' = (Q \cup \{\overline{q}\}, \Sigma, \Gamma, \delta', q_0, \Box, \{\overline{q}\})$ mit

$$\delta'(q, a) = \begin{cases} (\overline{q}, 0, N) & \text{falls } q \in F \text{ und } a = 1 \\ (\overline{q}, 1, N) & \text{falls } q \in F \text{ und } a = 0 \\ \delta(q, a) & \text{sonst} \end{cases}$$

   eine Turingmaschine, die $\overline{A}$ entscheidet, und zwar ebenso in polynomieller Zeit. Daher ist auch $\overline{A}$ in $\mathcal{P}$.

2. Folgender Algorithmus entscheidet dieses Problem:

   > Prüfe für je drei verschiedene Knoten des Graphen, ob sie untereinander verbunden sind.

   Da es höchstens n^3 verschiedene Tripel von Knoten eines Graphen mit insgesamt n Knoten gibt und der Test auf Verbundenheit dreier Knoten in polynomieller Zeit möglich ist, ist das Problem des Dreiecks in einem Graphen in $\mathcal{P}$.

Tutoraufgabe 1

Sei $\Sigma = \{a, b\}$. Bestimmen Sie alle Lösungen des Postschen Korrespondenzproblems

$$P_1 = \{(a, aaa), (abaaa, ab), (ab, b)\}$$

Lösung

Die Menge der Lösungen ist $\{(i_1 i_2 i_3 i_4)^n \,;\, n \in \mathbb{N}_0, i_1 = 2, i_2 = i_3 = 1, i_4 = 3\}$.

Tutoraufgabe 2

1. Ist NTIME_M für jede deterministische Turingmaschine M berechenbar? Begründung!

2. Sei $f : \mathbb{N}_0 \to \mathbb{N}_0$. Falls NTIME(f(n)) eine nichtentscheidbare Sprache enthält, dann ist f nicht berechenbar. Beweis!

3. Wir betrachten die Komplexitätsklasse $\mathcal{P}$. Dann gibt es für jede DTM M mit $L(M) \in \mathcal{P}$ ein Polynom p, so dass $\mathrm{TIME}_M(w) \leq p(|w|)$ für alle $w \in \Sigma^*$ gilt.

   Richtig oder Falsch? Begründung!

4. Ist jede in polynomieller Zeit berechenbare Funktion $f : \mathbb{N}_0 \to \mathbb{N}_0$ auch polynomiell beschränkt ($\exists$ Polynom p. $\forall n$. $f(n) \leq p(n)$)? Begründung!

Lösung

1. Eine deterministische Turingmaschine kann man als einen Spezialfall in der Klasse der nichtdeterministischen Turingmaschinen auffassen. Deshalb ist für M auch NTIME_M definiert. Im Unterschied zu $time_M$ ist allerdings NTIME_M eine totale Funktion auf Σ^*, der „Menge der Turingmaschinen“ (oder besser gesagt, der Menge der Codes aller Turingmaschinen).

   Sei nun A eine nicht entscheidbare Menge, die von einer DTM M akzeptiert wird. Wäre NTIME_M berechenbar, dann gäbe es eine DTM N, die NTIME_M berechnet, d. h. bei jeder Eingabe stoppt, also auch im Fall $\mathrm{NTIME}_M(w) = 0$. Dies würde aber bedeuten, dass man entscheiden könnte, wann M nicht hält, und damit wäre A entscheidbar. Widerspruch.

2. Sei A eine nicht entscheidbare Sprache, die von einer NTM M akzeptiert wird, so dass $\mathrm{NTIME}_M(w) \leq f(|w|)$ für alle w gilt. Von dem vorausgegangenen Beweis wissen wir, dass NTIME_M nicht berechenbar ist. Nun aber lernen wir, dass es keine, auch noch so große, berechenbare Funktion f gibt, mit der man NTIME_M abschätzen kann.

   Nehmen wir an, es gäbe eine berechenbare Funktion f mit obiger Eigenschaft. Dann könnten wir für ein beliebiges w den Wert $f(|w|)$ berechnen und die Turingmaschine M höchstens $f(|w|)$ Schritte rechnen lassen, um die Entscheidung über $w \in A$ zu erhalten. Widerspruch.

3. Falsch! $L(M) \in \mathcal{P}$ heißt lediglich, dass eine Turingmaschine M' existiert, die die Schrankenbedingung erfüllt. Es muss aber nicht $M = M'$ gelten.

4. Nein! Falls $f(n)$ als Strichzahl codiert wird, dann mag dies gelten. Es gilt aber nicht z.B. bei Binärdarstellung. Beispiel: $f(n) = 2^n$ f ist in linearer Zeit berechenbar, f ist aber nicht polynomiell beschränkt.

Tutoraufgabe 3

Wahr oder falsch? Begründen Sie Ihre Antwort!

1. Eine Sprache ist genau dann vom Typ 0, wenn sie rekursiv auflistbar ist.
2. Das folgende Problem ist entscheidbar:

   Gegeben: Eine deterministische Turingmaschine M.

   Problem: Schreibt M mit leerer Eingabe jemals ein nicht-$\square$ Symbol auf das Band?
3. Wenn eine Turingmaschine M bei einer Eingabe w das Band nie verändert, dann sagen wir, dass M ohne Speicherung arbeitet und schreiben dafür $OS(M, w)$. Wir definieren $OS = \{(v, w)\,;\, OS(M_v, w)\}$.

   Dann ist OS entscheidbar.
4. Die folgende Funktion $f : \mathbb{N}_0 \to \mathbb{N}_0$ ist berechenbar:

   $$f(x) = \begin{cases} 1 : P = NP \\ 0 : \text{sonst} \end{cases}$$

   Informieren Sie sich über die Probleme P und NP.

Lösung

1. Wahr! Begründung: Nach Satz der Vorlesung sind die Typ-0-Sprachen der Chomsky Hierarchie genau die von Turingmaschinen M akzeptierten Sprachen $L(M)$. Jede Sprache $L = L(M)$ ist semi-entscheidbar, weil deren „semi-charakteristische" Funktion χ'_L berechenbar ist durch eine Turingmaschine M', die 1 ausgibt, falls M hält, und ansonsten nicht hält.

   Nach Vorlesungslemma folgt aber aus der Semientscheidbarkeit von L, dass L auch rekursiv auflistbar ist.
2. Das Problem ist entscheidbar. Begründung:

   Die Anzahl der Zustände von T sei n. Sei $s > n$. Falls die Berechnung $(\epsilon, q_0, \epsilon) = k_0 \to_M k_1 \to_M \ldots \to_M k_s$ von T auf leerer Eingabe kein nicht-$\square$ Symbol enthält, dann gibt es $i < j$ und Konfigurationen $k_i = (\alpha_i, q_i, \beta_i)$, $k_j = (\alpha_j, q_j, \beta_j)$ mit $\alpha_i, \beta_i, \alpha_j, \beta_j \in \{\square\}^*$, so dass sich die Zustände wiederholen, d. h. $q_i = q_j$. In beiden Konfiguration k_i und k_j steht der Lesekopf über $\square$. Daraus folgt, dass sich das Schreibverhalten wiederholt, d.h. es wird nie ein eigentlicher Buchstabe, i.e. aus Σ, geschrieben.

   Wir entwerfen den folgenden Entscheidungsalgorithmus:

   Man setze die Turingmaschine M auf leerem Band an und führe Berechnungsschritte so lange aus, bis entweder ein nicht-$\square$ Symbol geschrieben wird oder sich ein Zustand wiederholt.

   Der Algorithmus terminiert.

3. Das Problem ist entscheidbar. Begründung:

   Für $w = \epsilon$ liegt der Fall der vorausgegangenen Teilaufgabe vor. Der dortige Algorithmus kann in diesem Fall übernommen werden. Wir betrachten im Folgenden den Fall $w \neq \epsilon$.

   Solange das Band nicht verändert wird, beschreibt jede berechnete Konfiguration eine der folgenden Situationen:

   1. Der Kontrollkopf steht über einem Leerfeld links von der Eingabe w.
   2. Der Kontrollkopf steht über einem Zeichen in w.
   3. Der Kontrollkopf steht über einem Leerfeld rechts von der Eingabe w.

   Sei n gleich der Anzahl der Zustände von M. Wir betrachten den Bereich B von $2(n+1)+|w|$ Feldern des Bandes, der sich aus den $n+1$ an die linke Seite der Eingabe w anschließenden Feldern, den Eingabefeldern von w und den $n+1$ an die rechte Seite der Eingabe w anschließenden Feldern zusammensetzt. Dann gibt es $n \cdot (2n+2+|w|)$ verschiedene Konfigurationen, in denen das Kontrollfeld in einem Zustand q auf einem Feld des Bereichs B steht.

   Nun berechnen wir s Schritte mit $s > n \cdot (2n+2+|w|)$.
   Wir betrachten den Fall, dass das Band nicht verändert wurde und der Kontrollkopf stets innerhalb von B geblieben ist. Dann muss sich eine Konfiguration wiederholt haben und die Berechnung periodisch werden. Das Band wird also nie geändert werden.
   Nun betrachten wir den Fall, dass das Band nicht verändert wurde und der Kontrollkopf außerhalb von B steht, z.B. links von B. Dann muss der letzten Konfiguration eine Konfiguration mit gleichem Zustand vorausgegangen sein, so dass zwischen den beiden Konfigurationen nur Leerfelder überschritten wurden. Dann aber wird der Kontrollkopf in allen folgenden Schritten periodisch nach links gehen und das Band wird nie geändert werden.
   Entsprechendes gilt für das Überschreiten von B nach rechts.
   Damit genügen s Berechnungsschritte mit M auf der Eingabe w, um zu entscheiden, ob das Band je geändert wird.

4. f ist berechenbar, weil f in jedem Fall eine konstante Funktion ist.