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

Sommersemester 2015
Lösungsblatt 9
29. Juni 2015

Theoretische Informatik

Hausaufgabe 1 (4 Punkte)

Gegeben sei das Alphabet $\Sigma = \{a, b, c, d\}$.

1. Definieren Sie einen Kellerautomaten $K_1 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$, der die Sprache $L_1 = \{ca^n db^n \,;\, n \in \mathbb{N}\}$ mit leerem Keller akzeptiert, so dass also $L_\epsilon(K_1) = L_1$ gilt! Geben Sie den Übergangsgraph Ihres Automaten K_1 an.

   Ist Ihr Automat K_1 deterministisch?

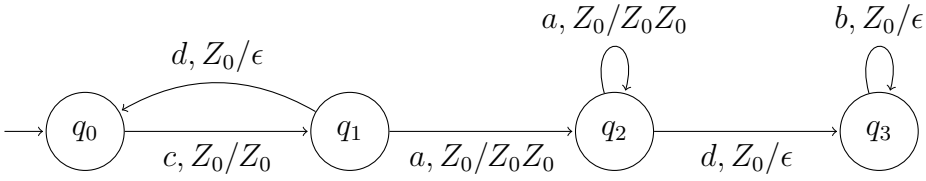
2. Wir betrachten für eine beliebige aber feste natürliche Zahl $k_0 \in \mathbb{N}$ (z.B. $k_0 = 2$) die Sprache
   $$L_{k_0} = \{c^{k_0} a^n d^{k_0} b^n \,;\, n \in \mathbb{N}\}\,.$$
   (a) Geben Sie für beliebiges $k_0 \geq 1$ ein Verfahren an zur Konstruktion einer Grammatik G_{k_0} in Chomsky-Normalform, so dass G_{k_0} die Sprache L_{k_0} erzeugt. Benützen Sie u.a. indizierte Nichtterminale U_i, V_i.

   (b) Erzeugen Sie für $k_0 = 2$ durch Anwendung Ihres Verfahrens eine konkrete Grammatik G_2, so dass $L(G_2) = L_2$ gilt.

3. Seien $k_0 \in \mathbb{N}$ beliebig aber fest und
   $$L = \{c^k a^n d^k b^n \,;\, k, n \in \mathbb{N}, k \leq k_0\}\,.$$
   Gibt es einen Kellerautomaten K, der L akzeptiert? Begründung!

Lösung

1. In graphischer Notation sieht K_1 wie folgt aus:



   Da kein spontaner Übergang definiert wurde, gilt $|\delta(q, x, Z_0)| + |\delta(q, \epsilon, Z_0)| \leq 1$ für alle $x \in \Sigma, q \in Q$. Damit ist K_1 deterministisch.

(1P)

2. (a) Wir definieren $G_{k_0} = (N, \Sigma, P, S)$ mit
$N = \{S, A, B, C, D, U_1, U_2 \ldots, U_{k_0}, V_1, V_2 \ldots, V_{k_0}\}$ und Produktionen aus P für alle $2 \leq i \leq k_0$ wie folgt:

$$\begin{array}{lll} S \to U_{k_0} V_{k_0}\,, & V_{k_0} \to AX\,, & X \to V_{k_0} B\,, \\ A \to a\,, & C \to c\,, & U_1 \to c\,, \\ B \to b\,, & D \to c\,, & V_1 \to d\,, \\ U_i \to CU_{i-1}\,, & V_i \to DV_{i-1}\,. & \end{array}$$

(b) Für $i = 2$ lautet die letzte Zeile: $U_2 \to CU_1$, $V_2 \to DV_1$.

(2P)

3. L ist für alle $k_0 \in \mathbb{N}$ kontextfrei, da alle L_i kontextfrei sind und L wegen

$$L = \bigcup_{i=0}^{k_0} L_i$$

eine endliche Vereinigung von kontextfreien Sprachen ist. Folglich gibt es auch einen entsprechenden Kellerautomat, der L akzeptiert.

(1P)

Hausaufgabe 2 (4 Punkte)

Sei $\Sigma = \{a, b, c\}$. Die Anzahl von Vorkommen eines Zeichens $x \in \Sigma$ in einem Wort $w \in \Sigma^*$ bezeichnen wir mit $\#_x(w)$. Für $w, u \in \Sigma^*$ heißt u *Präfix* von w, falls es ein $v \in \Sigma^*$ gibt, so dass $w = uv$ gilt.
Beispiel: ab ist Präfix von $w = aba$, aber ba ist nicht Präfix von w. Es gilt $\#_a(w) = 2$.

Wir definieren die Sprache L als die Menge aller Wörter w aus Σ^* mit der Eigenschaft, dass in jedem Präfix u von w die Anzahl der „öffnenden Klammern“ a größer oder gleich der Anzahl der „schließenden Klammern“ b ist, d.h.

$$L = \{w \in \Sigma^* \,;\, \text{für alle Präfixe } u \text{ von } w \text{ gilt } \#_b(u) \leq \#_a(u)\}\,.$$

1. Zeigen Sie mit Hilfe des Pumping Lemma, dass L nicht regulär ist.

2. Definieren Sie einen deterministischen Kellerautomaten $K = (Q, \Sigma, \Delta, \delta, q_0, Z_0, F)$, der die Sprache L mit Endzustand akzeptiert, so dass also $L(K) = L$ gilt!
Geben Sie dazu den Übergangsgraphen Ihres Automaten K an.

3. Gibt es einen deterministischen Kellerautomaten, der L mit leerem Keller akzeptiert? Begründung!

Lösung

1. Angenommen L sei regulär.

   Sei N eine Pumping-Lemma-Zahl für L und $z = a^N b^N$. Es gilt $z \in L$.
   Sei $z = uvw$, so dass $|uv| \leq N$, $v \neq \epsilon$ und für alle $i \in \mathbb{N}_0$ $uv^i w \in L$ gilt.

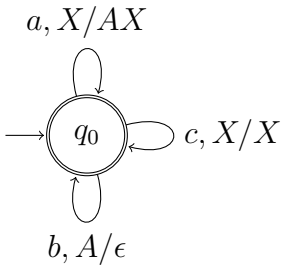
   Es folgt $uv \in a^+$, insbesondere $v = a^k$ für ein $0 < k \leq N$.

   Damit folgt für $i = 0$ $z_i := uv^i w = a^{N-k} b^N \in L$.

   Widerspruch, denn wegen $N - k < N$ gilt $\#_a(z_0) < \#_b(z_0)$, d.h. $z_0 \notin L$.

   (2P)

2. Seien $Q = \{q_0\}$, $\Delta = \{Z_0, A\}$, $F = \{q_0\}$.
   Für alle $X \in \Delta$:



   (1P)

3. Nein! L erfüllt nicht die Präfixbedingung, denn L enthält z.B. mit aab auch aa. (1P)

Hausaufgabe 3 (4 Punkte)

Konstruieren Sie für die folgenden Sprachen jeweils einen Kellerautomaten, der die Sprache erkennt.

(a) $L_1 = \{a^n b^{3n} \ ; \ n \in \mathbb{N}_0\}$

(b) $L_2 = \{wc\widehat{w} \ ; \ w \in \Sigma^*\}$, wobei $\widehat{w}$ das zu w gespiegelte Wort und $\Sigma = \{a, b\}$ ist.

(c) $L_3 = \{w\widehat{w} \ ; \ w \in \Sigma^*\}$, wobei $\widehat{w}$ das zu w gespiegelte Wort und $\Sigma = \{a, b\}$ ist.

Geben Sie – wenn möglich – einen deterministischen Kellerautomaten an.

Lösung

(a) Der Kellerautomat $M = (\{q_0, q_1, q_2\}, \{a, b, c\}, \{A, B, Z_0\}, \delta, q_0, Z_0, \{q_2\})$, wobei

$$\begin{aligned}
&\delta(q_0, a, *) = \{(q_1, AAA*)\}, \\
&\delta(q_1, a, *) = \{(q_1, AAA*)\}, \ \delta(q_1, b, A) = \{(q_2, \epsilon)\}, \\
&\delta(q_2, b, A) = \{(q_2, \epsilon)\}, \ \delta(q_2, \epsilon, Z_0) = \{(q_3, Z_0)\}
\end{aligned}$$

akzeptiert L durch Endzustände q_0 (leeres Wort) oder q_3. (2P)

(b) Der Kellerautomat $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{A, B, Z_0\}, \delta, q_0, Z_0, \{q_2\})$, wobei

$$\delta(q_0, a, *) = \{(q_0, A*)\},\ \delta(q_0, b, *) = \{(q_0, B*)\},$$
$$\delta(q_1, a, A) = \{q_1, \epsilon\},\ \delta(q_1, b, B) = \{(q_1, \epsilon)\},$$
$$\delta(q_0, c, *) = \{(q_1, *)\},\ \delta(q_1, \epsilon, Z_0) = \{(q_2, Z_0)\}.$$

akzeptiert L durch Endzustand q_2. (1P)

(c) Der Kellerautomat $M = (\{q_0, q_1\}, \{a, b\}, \{A, B, Z_0\}, \delta, q_0, Z_0)$, wobei

$$\delta(q_0, a, *) = \{(q_0, A*)\},\ \delta(q_0, b, *) = \{(q_0, B*)\},$$
$$\delta(q_1, a, A) = \{(q_1, \epsilon)\},\ \delta(q_1, b, B) = \{(q_1, \epsilon)\},$$
$$\delta(q_0, \epsilon, *) = \{(q_1, *)\},\ \delta(q_1, \epsilon, Z_0) = \{(q_1, \epsilon)\}$$

akzeptiert L durch leeren Keller ($*$ = symbolisiert das oberste Kellerzeichen). (1P)

Hausaufgabe 4 (4 Punkte)

Seien $L = \{a^i b^j c^k \,;\, i = j \text{ oder } j = k\}$ und $L' := \overline{L} \cap a^* b^* c^*$.

1. Zeigen Sie mit Hilfe von Ogden's Lemma, dass L' nicht kontextfrei ist.
2. Zeigen Sie, dass L nicht deterministisch kontextfrei ist.

Lösung

1. Es gilt $L' = \overline{L} \cap a^* b^* c^* = \{a^i b^j c^k \,;\, i \neq j \text{ und } j \neq k\}$.

   Wir nehmen an, dass L' kontextfrei ist und führen diese Annahme mit Hilfe Ogden's Lemma zum Widerspruch.

   Sei n wie in Ogden's Lemma. Wir setzen $z = a^{n+n!} b^n c^{n+n!}$. Offenbar gilt $z \in L'$. Wir markieren genau alle n b's. Seien $z = uvwxy$, vx enthalte mindestens eine Markierung, vwx enthalte höchstens n Markierungen, und für alle $i \geq 0$ gelte $z_i := uv^i wx^i y \in L'$.

   Es folgt, dass v Vorkommen von höchstens einem der drei Buchstaben a, b, c enthalten kann. Entsprechend kann x Vorkommen von höchstens einem der drei Buchstaben a, b, c enthalten. Allerdings muss entweder v oder x mindestens ein b enthalten.

   v enthalte mindestens ein b und vx enthalte p viele b's. Dann folgt mit geeignetem $y' \in c^*$, $s := n - p$ und $i - 1 = n!/p$ der Widerspruch

   $$z_i = a^{n+n!} b^s b^{pi} y' = a^{n+n!} b^{s+p} b^{p(i-1)} y' = a^{n+n!} b^{n+n!} y' \notin L' \,.$$

   Analog folgt der Widerspruch, falls x mindestens ein b enthält.

   (2P)

2. Offenbar ist L kontextfrei.

   Wir nehmen an, dass L sogar deterministisch kontextfrei ist. Damit ist das Komplement $\overline{L}$ ebenfalls deterministisch kontextfrei.

   Da $a^* b^* c^*$ regulär ist, folgt die Kontextfreiheit des Durchschnitts $L' := \overline{L} \cap a^* b^* c^*$. Widerspruch! (2P)

Hausaufgabe 5 (4 Punkte)

Für Zwecke dieser Aufgabe nennen wir einen deterministischen Kellerautomaten $K = (Q, \Sigma, \Delta, \delta, q_0, Z_0, F)$, dessen Übergangsfunktion δ so beschaffen ist, dass der Kellerinhalt nie verändert wird, einen ϵ-DFA. Sei $L(K)$ die Sprache, die von einem ϵ-DFA K mit Endzuständen akzeptiert wird.

Geben Sie ein direktes (nicht über ϵ-NFA) Verfahren an, das zu einem beliebigen ϵ-DFA K einen deterministischen endlichen Automaten A definiert, der die Sprache $L(K)$ erkennt!

Lösung

Da der Keller bei Berechnungen nie verändert wird, besteht der Kellerinhalt stets genau aus dem Zeichen Z_0. Entsprechend können die überflüssigen Argumente aus der Funktionalität der Übergangsfunktion δ entfernt und so der Funktionalität der Übergangsfunktion bei endlichen Automaten angepasst werden. Wenn wir auch die überflüssigen Symbole entfernen, dann können wir zu K eindeutig einen Automaten $K_e = (Q, \Sigma, \delta_e, q_0, F)$ definieren für alle $a \in \Sigma \cup \{\epsilon\}$ und $q, p \in Q$ mit

$$\delta_e(q, a) = p, \quad \text{falls} \quad \delta(q, a, Z_0) = \{(p, Z_0)\}\,.$$

Würde K_e keine ϵ-Übergänge enthalten, dann wäre K_e bereits im Wesentlichen ein deterministischer endlicher Automat, der $L(K)$ erkennt.
Falls K_e ausgehend von einem Zustand q ein Eingabezeichen $a \in \Sigma$ verarbeitet, dann geschieht dies, indem er eine, möglicherweise leere, Folge ϵ^n liest, dabei eine Folge von Zuständen durchläuft, und schließlich in einem Zustand ankommt, indem die Eingabe von a definiert ist. Anschließend wird a gelesen und in einen Zustand p übergegangen.
Entsprechend definieren wir eine Übergangsfunktion δ' wie folgt. Es sei $a \in \Sigma$ und n die kleinste Zahl aus $\mathbb{N}_0$, so dass $\hat{\delta_e}(q, \epsilon^n a)$ definiert ist, dann definieren wir

$$\delta'(q, a) = \hat{\delta_e}(q, \epsilon^n a)\,.$$

Wir erinnern hier an die Bemerkung in Blatt 5, Tutoraufgabe 1 zu den ϵ-Folgen und Ausdrücken, wenn ϵ als Buchstabe behandelt wird. δ' ist eine partiell definierte Funktion. Das Einlesen geschieht mit δ' vollständig und ϵ-frei. Damit können wir bereits den gesuchten Automaten wie folgt ansetzen.

$$A' = (Q, \Sigma, \delta', q_0', F')\,,$$

wobei eventuell $q_0' = \epsilon^n p_0$ für geeignetes n gesetzt wird. Wir müssen nun die Menge F' der Endzustände bestimmen.
Dazu müssen wir berücksichtigen, dass der Kellerautomat, nachdem er ein Zeichen gelesen hat, mit ϵ-Übergängen zu einem Endzustand laufen und akzeptieren kann. Dies bedeutet, dass dann der Zustand p, der bei Einlesen eines Zeichens angenommen wird, ebenfalls ein akzeptierender Zustand ist, sozusagen modulo nachfolgende ϵ. Deshalb definieren wir

$$F' = \{p \in Q \mid p \in F \vee (\exists n \in \mathbb{N})[\hat{\delta_e}(p, \epsilon^n) \in F]\,.$$

Schließlich totalisieren wir noch die partiell definierte Funktion δ' und erhalten als Erweiterung die Funktion δ'', indem wir für Argumentpaare, für die δ' nicht definiert ist, als Wert einen neuen Zustand f_0 festsetzen mit $Q' = Q \cup \{f_0\}$. Wir fassen das Ergebnis zusammen:

$$A = (Q', \Sigma, \delta'', q_0', F')\,. \qquad \text{(4P)}$$

Zusatzaufgabe 7 (wird nicht korrigiert)

Die Sprache P der Palindrome über dem Alphabet $\Sigma = \{0, 1\}$ ist gleich der Menge aller Wörter über Σ, die dieselbe Zeichenfolge ergeben, gleich ob man sie rückwärts oder vorwärts liest, d.h. $P = \{w \in \Sigma^* \,;\, w = w^R\}$.

1. Die Sprache P der Palindrome über dem Alphabet Σ ist kontextfrei. Zeigen Sie, dass P nicht regulär ist.

2. Sei $L \subseteq \Sigma^*$ regulär. Zeigen Sie die Regularität der folgenden Menge $L_{\frac{1}{2}P}$.

$$L_{\frac{1}{2}P} = \{w \in \Sigma^* \,;\, w^R w \in L\}\,.$$

Lösung

1. Wir nehmen an, dass P regulär ist und leiten mittels des Pumping-Lemmas einen Widerspruch her.

   Sei $n \in \mathbb{N}$ eine Pumping-Lemma-Zahl. Dann ist sicherlich $0^n 10^n \in P$. Es gibt also für z eine Zerlegung uvw mit $v \neq \epsilon$ und $|uv| \leq n$, so dass

$$(*) \quad uv^i w \in P \text{ für alle } i \in \mathbb{N}_0\,.$$

   Offensichtlich muss $uv = 0^k$ für $0 < k \leq n$ gelten, also $u = 0^i$ und $v = 0^j$ mit $i+j = k$ und $j > 0$. Dann ist aber $uv^2w = 0^i 0^{2j} 0^{n-k} 10^n \notin P$, denn $i + 2j + n - k > n$. Dies steht im Widerspruch zu $(*)$.

2. Die Idee zur Konstruktion eines Automaten zur Erkennung von $L_{\frac{1}{2}P}$ ist, dass man bei vorgelegtem w jeden Zustand testet, ob man mit w^R rückwärts zum Startzustand und gleichzeitig vorwärts mit w in einen Endzustand kommt. Man bildet hilfsweise Tupel (q_x, q_y) von Zuständen , die ein Paar von Markierungen bedeuten, wobei q_x den Weg zum Startzustand und q_y den Weg zu einem Endzustand verfolgen.

   Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DFA, der L erkennt. Wir definieren einen Produktautomaten $M^R = (Q \times Q, \Sigma, \delta', S', F')$ mit Startzuständen $S' = \{(q, q) \,;\, q \in Q\}$ und Endzuständen $F' = \{(q_0, q) \,;\, q \in F\}$. Dann ist die Übergangsfunktion δ' für alle $p, q \in Q$ und $x \in \Sigma$ wie folgt definiert.

$$\delta'((p, q), x) = \{(p', q') \,;\, \delta(p', x) = p \wedge \delta(q, x) = q'\}\,.$$

   Die von M^R erkannte Sprache ist gleich $L^R_{\frac{1}{2}P}$. Nach Satz der Vorlesung ist damit $L_{\frac{1}{2}P}$ regulär.

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

Wir betrachten den Beweis zu dem Satz der Vorlesung, in dem eine k-Band-Turingmaschine M durch eine normale Turingmaschine M' simuliert wird. Geben Sie eine Abschätzung für die Anzahl der Zustände an, die M' haben muss.

Lösung

Wir entwerfen den Zustandsraum von M' wie folgt.

- Zunächst muss natürlich der Zustand der simulierten Turingmaschine gespeichert werden. Sei Q die Zustandsmenge der simulierten Maschine.
- Es werden Zustände benötigt, um die Zeichen zu speichern, die jeweils über den $\star$ standen. Dafür wählen wir $Q_1 = \Gamma^k$.
- Für jede Spur (= jedes simulierte Band) merkt man einen Statuswert, der besagt wie weit man mit der Aktualisierung des Bandes schon gekommen ist. Es gibt vier mögliche Werte:

  U besagt, dass das Zeichen über dem $\star$ noch nicht gelesen wurde.

  R besagt, dass das Zeichen über dem $\star$ schon gelesen wurde.

  W besagt, dass das Zeichen über dem $\star$ gerade aktualisiert wurde. Dabei wurde der $\star$ gelöscht und muss im nächsten Schritt an neuer Position wieder geschrieben werden.

  M besagt, dass der $\star$ wieder geschrieben wurde.

  Damit ist $Q_2 = \{\mathbf{U}, \mathbf{R}, \mathbf{W}, \mathbf{M}\}^k$.

Mit der Zustandsmenge $Q \times Q_1 \times Q_2$ (also insgesamt $|Q| \cdot (4|\Gamma|)^k$ Zuständen) lässt sich nun die Turingmaschine M' beschreiben. Die Angabe der δ-Funktion sei dem geneigten Leser überlassen (Viele Fallunterscheidungen).
Im Detail könnte man natürlich noch optimieren und Zustände einsparen (z.B. sind einige der Zustände nicht erreichbar). Allerdings bleibt in jedem Fall der Term $|\Gamma|^k$ erhalten, da man sich mindestens k Zeichen aus Γ merken muss.

Vorbereitung 2

Seien $\Sigma = \{a_1, a_2, \ldots, a_n\}$ ein beliebiges n-elementiges Alphabet und $\Sigma' = \Sigma \cup \{\#\}$.
Geben Sie eine Turingmaschine $M = (Q, \Sigma', \Gamma, \delta, q_0, \square, F)$ mit höchstens 5 Zuständen an, die bei leerer Eingabe das Alphabet Σ in der Form $\#a_1\#a_2 \ldots \#a_n$ auf das Band schreibt und mit dem Kopf auf dem letzten, rechtsstehenden Zeichen der Ausgabe anhält.

Lösung

Sei $M = (\{q_0, q_L, q_A, q_f\}, \Sigma', \Sigma' \cup \{\square\}, \delta, q_0, \square, \{q_f\})$.

Übergang		Bereich	Kommentar
$\delta(q_0, \square)$	$\rightarrow (q_L, \square, L)$		Übergang in Linkslauf an Wortanfang.
$\delta(q_0, a_i)$	$\rightarrow (q_0, a_{i+1}, R)$	$i < n$	„Shift“ des Alphabets.
$\delta(q_0, \#)$	$\rightarrow (q_0, \#, R)$		„Shift“ des Alphabets.
$\delta(q_L, a_n)$	$\rightarrow (q_f, a_n, N)$		Ende.
$\delta(q_L, a_i)$	$\rightarrow (q_L, a_i, L)$	$i < n$	nach links gehen
$\delta(q_L, \#)$	$\rightarrow (q_L, \#, L)$		nach links gehen
$\delta(q_L, \square)$	$\rightarrow (q_A, a_1, L)$		Schreiben am Wortanfang.
$\delta(q_A, \square)$	$\rightarrow (q_A, \#, R)$		Schreiben von #.
$\delta(q_A, a_1)$	$\rightarrow (q_0, a_1, R)$		Zum Start.

Vorbereitung 3

Wahr oder falsch? Begründen Sie Ihre Antworten:

1. Jede unentscheidbare Sprache enthält eine entscheidbare Teilmenge.
2. Jede Teilmenge einer entscheidbaren Sprache ist entscheidbar.
3. Für jede unentscheidbare Sprache A gibt es eine echte Obermenge, die ebenfalls unentscheidbar ist.
4. Aus „A entscheidbar“ und „$A \cap B$ entscheidbar“ folgt „B entscheidbar“.

Lösung

Eine Menge $A \subseteq \Sigma^*$ heißt entscheidbar, falls die charakteristische Funktion χ_A total auf Σ^* und berechenbar ist.
Dann ergeben sich die folgenden Antworten und Begründungen.

1. Ja, denn jede Sprache $L \subseteq \Sigma^*$ enthält eine endliche Teilmenge, und jede endliche Teilmenge von Σ^* ist entscheidbar.

   Bemerkung: Eine z. B. einelementige Menge $\{a\}$ ist zwar entscheidbar. Dies muss aber nicht bedeuten, dass man dieses eine Element $a \in \Sigma^*$ kennt bzw. zu konstruieren in der Lage ist. Wir wissen lediglich die abstrakte Existenz dieses a bzw. die Existenz eines Algorithmus, der für vorgelegtes w den Wert der charakteristischen Funktion berechnet, d. h. prüft, ob $w = a$ gilt, wobei der Vergleich $w = a$ für Wörter natürlich mit abbrechendem Algorithmus berechenbar ist.

2. Nein, denn $\{0,1\}^*$ ist entscheidbar, aber das allgemeine Halteproblem $H \subseteq \{0,1\}^*$ ist nicht entscheidbar.

3. Ja, denn für ein $w \notin A$ (und das existiert immer, denn Σ^* ist entscheidbar) ist $A \cup \{w\}$ ebenfalls unentscheidbar, falls A unentscheidbar ist.

4. Nein. Gegenbeispiel: $\varnothing$ und $\varnothing \cap H$ sind entscheidbar, weil beide Mengen endlich sind. H (das allgemeine Halteproblem) ist aber nicht entscheidbar.

Vorbereitung 4

In einem Tresor liegt eine Liste mit 6-stelligen TAN-Nummern. Der Schlüssel zum Öffnen des Tresors ist verloren gegangen und es gibt keine andere Möglichkeit, den Tresor zu öffnen.

Sei A die Menge der Primzahlen, die auf der TAN-Liste vorkommen. Ist $A \subseteq \mathbb{N}$ entscheidbar? Begründung!

Lösung

A ist entscheidbar, weil A endlich ist.

Entscheidbarkeit heißt nicht, dass irgend jemand in der Lage sein muss, die Entscheidung zu treffen.

Man muss nur nachweisen, dass es einen Entscheidungsalgorithmus gibt.

Die Begriffe Berechenbarkeit und Entscheidbarkeit sind Strukturbegriffe.

Bemerkung:

Eine z. B. einelementige Menge $\{a\}$ ist zwar entscheidbar. Dies muss aber nicht bedeuten, dass man dieses eine Element $a \in \Sigma^*$ kennt bzw. zu konstruieren in der Lage ist.

Wir wissen lediglich die abstrakte Existenz dieses a bzw. die Existenz eines Algorithmus, der für vorgelegtes w den Wert der charakteristischen Funktion berechnet, d. h. prüft, ob $w = a$ gilt, wobei der Vergleich $w = a$ für Wörter natürlich mit abbrechendem Algorithmus berechenbar ist.

Vorbereitung 5

Man zeige:

1. Sei $\Sigma = \{0, 1\}$ und $f : \Sigma^* \to \Sigma^*$ eine beliebige (möglicherweise partielle) Funktion. Der Graph von f ist die Relation $G_f = \{(v, w) \in \Sigma^* \times \Sigma^* \,;\, f(v) = w\}$.

   Wenn G_f entscheidbar ist, dann ist f berechenbar.

2. Gegeben sei eine berechenbare Auflistung (Codierung) aller Turingmaschinen, die jedem Wort $w \in \{0, 1\}^*$ eine Turingmaschine M_w zuordnet. Dann ist die Sprache $L = \{w \mid L(M_w) \text{ ist rekursiv aufzählbar}\}$ entscheidbar.

Lösung

1. Ausgehend von einem Entscheidungsverfahren für G_f erhält man den folgenden Algorithmus für f:

   > Gegeben ein $v \in \Sigma^*$, zähle alle $w \in \Sigma^*$ nacheinander auf und überprüfe jeweils, ob $(v, w) \in G_f$. Falls ja, dann gib w als Ergebnis zurück. Falls nein, dann fahre mit dem nächsten Wort fort.

   Falls $f(v) \neq \perp$, so wird der Funktionswert irgendwann gefunden. Andernfalls terminiert das Verfahren nicht, was ja genau das geforderte Verhalten ist.

2. Eine Sprache ist semi-entscheidbar, wenn sie die von einer Turingmaschine akzeptierte Sprache ist.

   Definitionsgemäß ist $L(M_w)$ die von M_w akzeptierte Sprache. Also ist $L(M_w)$ semi-entscheidbar für alle w. Damit folgt $L = \{0,1\}^*$, d. h., L ist trivialerweise entscheidbar.

Tutoraufgabe 1

Zeigen Sie, dass jede (deterministische) Turingmaschine durch einen Queue-Automaten (siehe HA 4 von Blatt 8) simuliert werden kann.

Lösung

Sei $M = (Q, \Sigma, \Gamma, q_0, \delta, \square, F)$ die Turingmaschine, die wir durch einen Queue-Auto-maten A simulieren wollen. Wir nehmen an, dass $Q \cap \Gamma = \emptyset$.
Die Konstruktionsidee ist nun, dass wir eine Konfiguration (v, q, w) der Turingmaschine als Wort $vqw\#$ in der Queue ablegen. Der QA kann die Queue rotieren, indem er ein Queue-Zeichen liest, und danach wieder ans Ende der Queue anhängt. So lässt sich die Queue flexibel manipulieren. Für jeden Schritt der Turingmaschine muss allerdings die Queue einmal komplett durchlaufen werden.
Unser Queue-Automat A hat die Zustandsmenge

$$Q' = \{\mathit{ff}, \mathit{start}\} \cup \{\mathit{find}_x \mid x \in \Gamma\} \cup \{\mathit{step}_x^q \mid x \in \Gamma, q \in Q\}$$

und die folgenden schematischen Übergänge, parametrisiert für alle $x, y \in \Gamma$ und $q \in Q$:

$$\begin{aligned}
\delta'(\mathit{start}, \epsilon, x) &\to (\mathit{find}_x, x) && \text{(zum Kopf vorlaufen, dabei}\\
\delta'(\mathit{find}_x, \epsilon, y) &\to (\mathit{find}_y, x) && \text{jeweils letztes Zeichen merken)}\\
\delta'(\mathit{find}_x, \epsilon, q) &\to (\mathit{step}_x^q, \epsilon) && \text{(zusätzlich den Zustand merken)}\\
\delta'(\mathit{step}_x^q, \epsilon, y) &\to \begin{cases} (\mathit{ff}, xq'z) & \text{falls } \delta(q,y) = (q', z, N)\\ (\mathit{ff}, xzq') & \text{falls } \delta(q,y) = (q', z, R)\\ (\mathit{ff}, q'xz) & \text{falls } \delta(q,y) = (q', z, L) \end{cases} && \text{(Schritt)}\\
\delta'(\mathit{ff}, \epsilon, x) &\to (\mathit{ff}, x) && \text{(Vorspulen bis zum \#)}\\
\delta'(\mathit{ff}, \epsilon, \#) &\to (\mathit{start}, \#) && \text{(von vorn beginnen)}\\
\delta'(\mathit{start}, \epsilon, q) &\to (\mathit{step}_\square^q, \epsilon) && \text{(Sonderfälle für den Rand)}\\
\delta'(\mathit{step}_x^q, \epsilon, \#) &\to (\mathit{start}, xq\square\#)
\end{aligned}$$

Nun gilt (hier ohne Beweis):

$$(\mathit{start}, \epsilon, vqw\#) \to_A^* (\mathit{start}, \epsilon, v'q'w'\#) \iff (v, q, w) \to_M^* (v', q', w')$$

Um dafür zu sorgen dass auch wirklich $L(A) = L(M)$ gilt, muss man den Queue-Automaten noch so erweitern, dass er am Anfang das Eingabewort in die Queue schreibt, und bei Erreichen eines Endzustands der TM die Queue komplett entleert.

Tutoraufgabe 2

Wir bezeichnen mit TM_k solche Einband-Turingmaschinen, die jede Zelle des Bandes höchstens k-mal ändern dürfen. Dabei gelten nur Übergänge $\delta(q, x) = (q', y, X)$ mit $x \neq y$ als Änderungen einer Zelle des Bandes (mit $X \in \{N, R, L\}$).

1. Zeigen Sie, dass die Turingmaschinen TM_2 äquivalent zu herkömmlichen Turingmaschinen sind. Benutzen Sie soviel Band wie nötig.

2. Zeigen Sie, dass auch die Turingmaschinen TM_1 äquivalent zu herkömmlichen Turingmaschinen sind. Sie dürfen dabei die Resultate der ersten Teilaufgabe verwenden.

Sie müssen keine expliziten Konstruktionen angeben. Es genügen informelle, aber dennoch vollständige und genaue Beschreibungen.

Lösung

1. Sei A eine herkömmliche Turingmaschine und A' eine TM_2.

   Allgemein gilt die folgende Beobachtung: Zu jedem Zeitpunkt steht nur endlich viel Information auf dem Band einer Turingmaschine, und der Rest des Bandes ist mit Leerzeichen gefüllt. Wir können daher den informationstragenden Teil des Bandes durch je eine eindeutige Markierung umranden.

   Wir simulieren nun einen Schritt von A in A' wie folgt. Zunächst kopieren wir den gesamten Bandinhalt zwischen den Randmarkierungen nach rechts. Kopieren erfolgt zeichenweise, und jedes kopierte Zeichen wird markiert. Daher wird jede Zelle zweimal geändert, beim ersten Schreiben (als Ziel einer Kopie) und beim Markieren (als Quelle einer Kopie). Außerdem markieren wir uns die Position des Lese-/Schreibkopfes von A auf dem Band von A'. Beim Kopieren der Zellen, die direkt benachbart zur gespeicherten Position des Lese-/Schreibkopfes liegen, werden nur die Ergebnisse entsprechend der aktuell auszuführenden Transition von A geschrieben.

2. Analog zur Simulation einer Turingmaschine durch eine TM_2 erfolgt auch die Simulation einer Turingmaschine A durch eine TM_1 A', allerdings mit dem Unterschied, dass wir nun jede Bandzelle von A durch zwei Zellen in A' simulieren. In der ersten Zelle steht das Zeichen von A (oder die gespeicherte Position des Lese-/Schreibkopfes), und die zweite Zelle wird zur Markierung beim Kopieren benutzt, bleibt aber anfangs unbeschrieben (und deshalb darf man annehmen, dass dort das Leerzeichen steht). Da die Eingabe nicht dem Zwei-Zellen-Format entspricht, muss sie zunächst einmal in dieses Format kopiert werden, wobei die ursprünglichen Eingabezeichen zur Markierung einmal überschrieben werden.

Tutoraufgabe 3

Sei $\Sigma = \{a, b, *\}$, $\Gamma = \Sigma \cup \{\square\}$ und $Q = \{q_0, q_1, q_2, q_f\}$. Wir betrachten die Turingmaschine $N = (Q, \Sigma, \Gamma, \delta, q_0, \square, \{q_f\})$ mit der Übergangsfunktion

$$
\begin{array}{lcllcl}
\delta(q_0, a) & = & \{(q_0, a, R)\}, & \delta(q_0, b) & = & \{(q_0, b, R)\}, \\
\delta(q_0, *) & = & \{(q_0, a, R)\}, & \delta(q_0, *) & = & \{(q_0, b, R)\}, \\
\delta(q_0, \square) & = & \{(q_1, \square, L)\}, & & & \\
\delta(q_1, a) & = & \{(q_2, \square, L)\}, & \delta(q_2, b) & = & \{(q_1, \square, L)\}, \\
\delta(q_1, \square) & = & \{(q_f, \square, N)\}. & & &
\end{array}
$$

1. Geben Sie eine deterministische Turingmaschine M an, die die Sprache $L(N)$ erkennt.

2. Beschreiben Sie ein allgemeines Verfahren, das zu jeder beliebigen nichtdeterministischen Turingmaschine N eine äquivalente deterministische Turingmaschine M liefert, d. h., so dass $L(N) = L(M)$ gilt.

Lösung

Wir erkennen an der Definition der Menge $\{q_f\}$ der Endzustände von N, dass die akzeptierte Sprache $L(N)$ mit Endzuständen akzeptiert wird. So jedenfalls war die Konvention für die Schreibweise in der Vorlesung.

1. Im Regelfall haben spezielle Probleme beste Lösungen, die durch Anwendung allgemeiner Verfahren nicht gefunden werden können. Meist greift dann die Empfehlung, das Gehirn zum selbständigen Denken zu benutzen.

   Abseits allgemeiner Verfahren zur Konstruktion deterministischer Varianten von nichtdeterministischen Turingmaschinen kann man die Frage stellen, wie man in diesem speziellen Fall die von N akzeptierte Sprache knapp beschreiben und damit begreifen kann.

   Offenbar lassen sich die Berechnungen in 2 Phasen einteilen. In der ersten Phase wird der Lese/Schreibkopf nach rechts bewegt bis zum ersten Blank rechts nach der Eingabe. Dabei werden alle Zeichen $*$ nichtdeterministisch in a oder b gewandelt. In der zweiten Phase wird der Lese/Schreibkopf nach links bewegt, ggf. bis zum ersten Blank links vor der Eingabe, und zwar dann, wenn abwechselnd a und b auf dem Band stehen. Andernfalls stoppt die Maschine in der zweiten Phase. Schließlich geht die Maschine in den Endzustand, falls das letzte gelesene Zeichen auf der äußersten linken Seite ein b war. Die Maschine geht auch dann in den Endzustand, wenn das Eingabeband leer ist.

   Den i-ten Buchstaben eines Wortes w bezeichnen wir nun als w_i. Wir sagen, dass ein Wort $w \in \Sigma^*$ die Eigenschaft P hat, i. Z. $P(w)$, falls
   1. die Länge von w geradzahlig ist,
   2. $w_{2i} \in \{a, *\}$ und
   3. $w_{2i-1} \in \{b, *\}$
   gilt für alle $i \geq 1$.

   Dann gilt $L(N) = \{w \in \Sigma^* ; P(w)\}$.

$L(N)$ ist sogar regulär. Eine deterministische Turingmaschine braucht das Band also lediglich zu lesen, um die Sprache zu erkennen.

$$\begin{aligned}
\delta(q_0, \Box) &= \{(q_f, \Box, N)\}, & \delta(q_0, b) &= \{(q_1, b, R)\}, \\
\delta(q_0, *) &= \{(q_1, *, R)\}, & & \\
\delta(q_1, a) &= \{(q_2, a, R)\}, & \delta(q_1, *) &= \{(q_2, *, R)\}, \\
\delta(q_2, b) &= \{(q_1, b, R)\}, & \delta(q_2, *) &= \{(q_1, *, R)\}, \\
\delta(q_2, \Box) &= \{(q_f, \Box, N)\}. & &
\end{aligned}$$

2. Wir betrachten nun die grundsätzliche Seite des Problems. Gegeben sei eine nichtdeterministische Turingmaschine $N = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ mit durch Endzustände akzeptierter Sprache $L(N)$. Wir konstruieren eine deterministische Turingmaschine $M = (Q', \Sigma', \Gamma', \delta', q_0', \Box', F')$ mit durch Endzustände akzeptierter Sprache $L(M)$, so dass $L(M) = L(N)$ gilt.

   Beide Sprachen müssen als Teilmengen von Σ^* aufgefasst werden. Dies folgt aus dem Äquivalenzkonzept von Turingmaschinen. Wir setzen also sofort $\Sigma' = \Sigma$.

   Die Konstruktionsidee für M ist, deterministisch den Konfigurationsbaum $K(w)$, der die Berechnung des Ergebnisses bei Eingabe eines Wortes $w \in \Sigma^*$ in die Turingmaschine N darstellt, schichtweise zu erzeugen (nach BF Strategie) und zu überprüfen, ob N einen Endzustand erreicht, in welchem Fall dann M in einen Endzustand gehen muss. Die im Baum $K(w)$ dargestellten Konfigurationen von N werden dabei linear auf das Band von M geschrieben, so dass jede Schicht rechts an die vorausgegangene Schicht anschließt.

   Eine Konfiguration der Turingmaschine N ist ein Tripel $\alpha q \beta$, wobei $\alpha, \beta \in \Gamma^*$ und $q \in Q$. Um eine Konfiguration von N auf das Band von M schreiben zu können, müssen die Zustände aus Q als Symbole in Γ' kodiert sein und es muss $\Gamma \subseteq \Gamma'$ gelten. Für die Trennung der Kodierungen der Konfigurationen auf dem Band von M verwenden wir ein gesondertes Trennzeichen $\# \in \Gamma'$. Es dürfen $\Box$ und $\Box'$ nicht gleichgesetzt werden.

   Die Beschreibung einer komplexen Turingmaschine erfordert, genau wie jede Art von guter Programmierung, eine Strukturierung in Unterprogramme. Wir werden nun das Gesamtprogramm für M beschreiben relativ zu Unterprogrammen bzw. speziellen Turingmaschinen zur Erledigung spezieller Aufgaben, wobei die Unterprogramme aber nur spezifiziert und nicht implementiert werden.

   - Im Zustand q_0' von M wird eine Maschine M_E aufgerufen, die das Eingabewort w auf dem sonst leeren Band von M als Konfiguration $\epsilon q_0 w$ von N auf das Band schreibt. Das Trennzeichen # wird links vor die Konfiguration geschrieben und der Kopf auf # gesetzt. Der Zustand q_1' wird angenommen.
   - Im Zustand q_1' von M wird eine Maschine M_1 aufgerufen, die die anliegende Konfiguration von N darauf überprüft, ob sie einen Endzustand von N enthält. Wenn ja, dann geht M in einen eigenen Endzustand. Wenn nein, dann geht M in den Zustand q_2'.
   - Im Zustand q_2' von M wird eine Maschine M_2 aufgerufen, die zunächst alle endlich vielen möglichen, Folgekonfigurationen der anliegenden Konfiguration

rechts ans Bandende, durch # getrennt, anfügt und die von links erste Konfiguration löscht. Der Kopf wird auf das erste # gesetzt. Der Zustand q_1' wird angenommen.

M geht bei Eingabe von $w \in \Sigma^*$ genau dann in einen Endzustand, wenn es für N bei gleicher Eingabe einen mögliche Berechnung gibt, die im Endzustand endet.

Tutoraufgabe 4

Eine Menge natürlicher Zahlen läßt sich als Teilmenge von Σ^+ über einem einelementigen Alphabet $\Sigma = \{|\}$ kodieren. Entsprechend werden wir Begriffe für formale Sprachen auf Mengen natürlicher Zahlen anwenden.
Wir betrachten die Menge $G = \{n \in \mathbb{N}\,;\, n \neq 1, (\neg\exists\,\text{Primzahlen}^1\ x, y)[2n = x + y]\,\}$.

1. Geben Sie eine knappe Begründung, warum G entscheidbar ist!

2. Vermutlich werden Sie keine der Aussagen beweisen können, ob G leer ist oder nicht, denn Sie müssten dazu die Goldbachsche Vermutung beweisen oder widerlegen. Warum können Sie trotzdem zeigen, dass für G das Leerheitsproblem entscheidbar ist?

Lösung

1. G ist definitionsgemäß als Teilmenge von $\mathbb{N}_0$ genau dann entscheidbar, wenn es eine berechenbare charakteristische Funktion $\chi_G : \mathbb{N}_0 \to \{0, 1\}$ gibt.

   Da Primzahlen aufgezählt werden können, und zwar nach Größe sortiert, ausgehend von 2, können wir für jedes n effektiv die endliche Liste aller Primzahlen berechnen, die kleiner oder gleich n sind. Nun können effektiv alle Paare von Elementen dieser Liste geprüft werden, ob deren Summe $2n$ ergibt. Je nach dem gilt dann $n \in G$, d. h. $\chi_G(n) = 1$, oder $n \notin G$, d. h. $\chi_G(n) = 0$.

2. Bisher haben wir meist unendliche Klassen bzw. Mengen betrachtet und mit einem gewissen Prädikat ein Problem definiert. Beispielsweise haben wir für die Klasse bzw. die Menge von kontextfreien Sprachen über einem Alphabet Σ das Leerheitsprädikat betrachtet. Wir konnten nachweisen, dass dieses Prädikat berechenbar ist.

   In dem Fall der Goldbachmenge G haben wir eine Klasse von Mengen vor uns, die nur aus einem einzigen Element besteht. Die Klasse ist also endlich. Jedes Prädikat über einer endlichen Menge ist aber berechenbar.

   Das Leerheitsprädikat $L(x)$ besitzt eine der 2 möglichen charakteristischen Funktionen, nämlich $\chi_1(G) = 0$ oder $\chi_2(G) = 1$. Wir wissen nicht, welche von beiden zu dem Prädikat gehört. Beide Funktionen sind aber trivialerweise berechenbar. Also existiert eine berechenbare charakteristische Funktion. Mithin ist das Leerheitsproblem entscheidbar.

[1] 1 ist keine Primzahl