Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
L. Hübschle-Schneider, T. Maier

7. Übungsblatt zu Theoretische Grundlagen der Informatik im WS 2015/16

http://algo2.iti.kit.edu/TGI2015.php
{sanders,huebschle,t.maier}@kit.edu

Musterlösungen

Wir verwenden in der Übung die Konvention, dass eine Turingmaschine die Eingabe genau dann akzeptiert, wenn sie in einem Endzustand hält.

Aufgabe 1 (*Turingmaschinenkonstruktion,* $2 + 3 + 2$ *Punkte*)

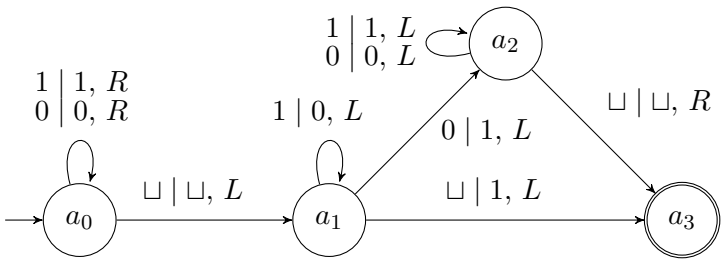
Das Ziel dieser Aufgabe ist die Konstruktion einer deterministischen Turingmaschine für Binärzahladdition. Die Maschine soll führende Nullen in der Eingabe erlauben, die Ausgabe darf jedoch keine führenden Nullen enthalten. Auf den Vorlesungsfolien ist bereits eine Maschine für Inkrementation einer Binärzahl gegeben, wir haben diese in einfacherer Form unten repliziert. Erklären Sie für die von Ihnen konstruierten Maschinen jeweils, wie diese funktionieren und was welcher Zustand tut. Erweitern Sie das Bandalphabet *nicht*.

a) Entwerfen Sie eine Turingmaschine, die eine binärkodierte natürliche Zahl dekrementiert. Die Maschine soll akzeptierend halten, wenn das Ergebnis eine natürliche Zahl ($\mathbb{N}_0$) ist, und nichtakzeptierend auf leerem Band, wenn die Eingabezahl 0 ist. Die Ausgabe darf keine führenden Nullen aufweisen. Verwenden Sie maximal 6 Zustände, 5 reichen aus. $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \sqcup\}$.
Update 16.12.: Wir wurden darauf hingewiesen, dass die Behandlung aller Randfälle nicht mit 5 Zuständen möglich ist. Die Maximalzustandszahl wird darum auf 7 Zustände erhöht, bitte melden Sie sich bei Ihrem Tutor.

b) Kombinieren Sie die gegebene Inkrementierungsmaschine mit Ihrer Dekrementierungsmaschine, sodass diese bei Eingabe $n_1 X n_2$, wobei X ein Trennsymbol ist und n_1 und n_2 binärkodierte natürliche Zahlen (mit Null, $\mathbb{N}_0$) sind. Addieren Sie n_2 auf n_1. Sie dürfen annehmen, dass mindestens eine der beiden Zahlen nicht Null ist. Am Ende soll das Band genau die Summe der beiden Zahlen, nicht aber ein Trennsymbol oder führende Nullen enthalten, und die Maschine in einem Endzustand halten. Wie können Sie mit Randfällen (beispielsweise wenn eine der beiden Zahlen 0 ist) umgehen? Verwenden Sie maximal 11 Zustände, 8 reichen aus. $\Sigma = \{0, 1, X\}$, $\Gamma = \{0, 1, X, \sqcup\}$.
Tipp: Sie können während der Berechnung beliebig weitere Trennsymbole einfügen, beispielsweise wenn die zweite Zahl durch Dekrementierungen kürzer wird.

c) Simulieren Sie Ihre Turingmaschine aus Aufgabenteil b) auf der Eingabe $011X10$.

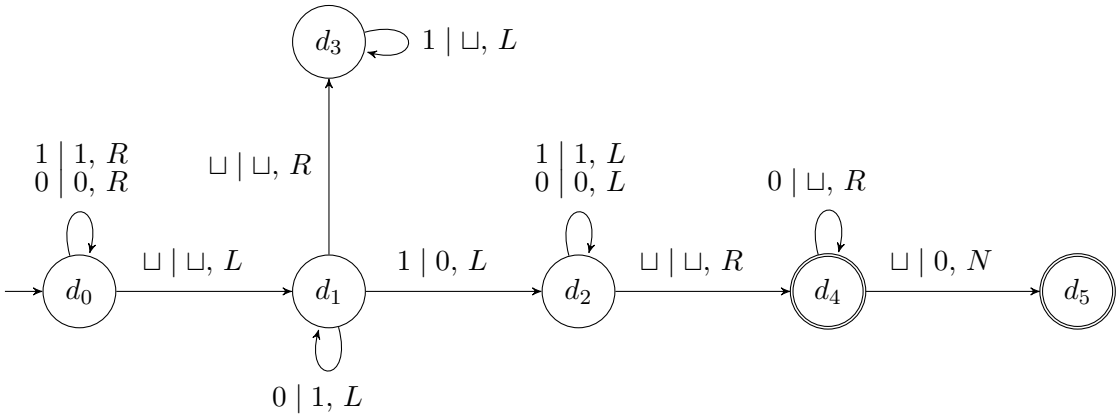
Für jeden weiteren Zustand wird ein Punkt abgezogen.



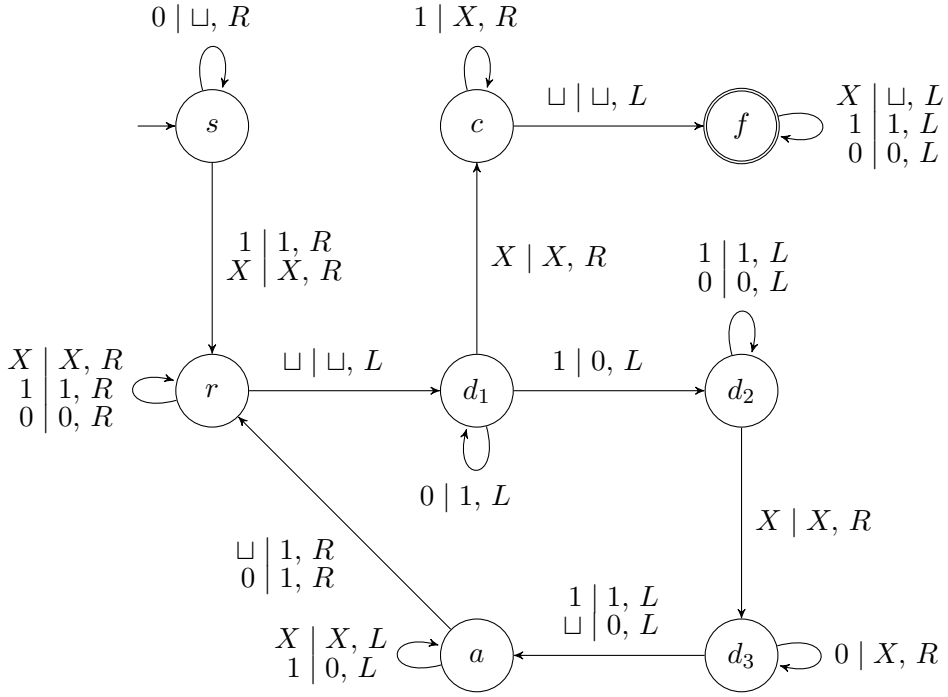
Musterlösung:

a) Sei $\mathcal{M}_{dec} = (\{d_0, \ldots, d_4\}, \{0,1\}, \{0,1,\sqcup\}, \delta_{dec}, d_0, \{d_4\})$. Die Übergangsrelation δ_{dec} ist gegeben durch die untenstehende Zeichnung.

Die Maschine funktioniert prinzipiell sehr ähnlich zur Inkrementierungsmaschine, benötigt aber eine Sonderbehandlung für den Fall, dass das Band nur aus Nullen besteht. Zuerst läuft sie ganz nach rechts, und subtrahiert dann in d_1 eins von der Zahl, verbleibt bei Übertrag in d_1 und geht nach d_2 über, falls nur noch eventuelle führende Nullen gelöscht werden müssen. Falls nur Nullen auf dem Band standen, geht die Maschine von d_1 nach d_3 über und löscht das Band. Sie hält dann implizit und nichtakzeptierend in d_3. Ansonsten hält sie akzeptierend in d_4. Sollte die Eingabe 1 gewesen sein, wird die nun fehlende Null durch den Übergang nach d_5 wieder eingefügt.



b) Sei $\mathcal{M}_{add} = (\{s, d_1, d_2, d_3, c, f, a\}, \{0,1,X\}, \{0,1,X,\sqcup\}, \delta_{add}, s, \{f\})$. Der untenstehende Zustandsübergangsgraph spezifiziert δ_{add}.



Die Maschine läuft nach rechts, dekrementiert die zweite und inkrementiert die erste Zahl. Dies wird wiederholt, bis die zweite Zahl Null ist.

s löscht führende Nullen in der ersten Zahl und geht dann nach r über.

r läuft beide Zahlen nach rechts ab.

d_1 entspricht d_1 aus Aufgabenteil a), allerdings steht hier links der zweiten Zahl kein $\sqcup$ sondern das Trennsymbol X. Der Übergang nach c behandelt denn Fall, dass die zweite Zahl Null ist und die Addition abgeschlossen wurde.

d_2 läuft die hintere Zahl bis zu ihrem Anfang

nach vorne und geht dann nach d_3 über.

d_3 entfernt führende Nullen der zweiten Zahl und ersetzt diese durch weitere Trennzeichen. Falls die zweite Zahl null ist, wird im Übergang nach a wieder eine einzelne Null geschrieben. Dieser Fall wird im nächsten Durchlauf in d_1 behandelt.

a übernimmt die Addition – hierfür ist nur ein Zustand nötig! Die erste gefundene Null wird durch eine Eins ersetzt, alle vorherigen Einsen durch Nullen. Falls das Wort nur aus Einsen bestand, wird eine führende Eins angehängt, da die Zahl nun eine Stelle mehr benötigt.

c löscht die zweite Zahl, falls diese Null war. Dazu werden zuerst alle eben geschriebenen Einsen (es können mehrere sein, da führende Nullen in der Eingabe erlaubt sind) durch Trennsymbole ersetzt. Am Wortende folgt der Übergang nach f.

f löscht alle Trennsymbole und hält dann akzeptierend vor dem Beginn der Ausgabe. Die Addition ist fertig.

c)

$(s)011X10$	$11X(a)X1$	$100X(d2)X0$	$101XXX(d_1)0$
$(s)11X10$	$11(a)XX1$	$100XX(d_3)0$	$101XX(d_1)X1$
$1(r)1X10$	$1(a)1XX1$	$100XXX(d_3)\sqcup$	$101XXX(c)1$
$11(r)X10$	$(a)10XX1$	$100XX(a)X0$	$101XXXX(c)\sqcup$
$11X(r)10$	$(a)\sqcup 00XX1$	$100X(a)XX0$	$101XXX(f)X$
$11X1(r)0$	$1(r)00XX1$	$100(a)XXX0$	$101XX(f)X$
$11X10(r)\sqcup$	$10(r)0XX1$	$10(a)0XXX0$	$101X(f)X$
$11X1(d_1)0$	$100(r)XX1$	$101(r)XXX0$	$101(f)X$
$11X(d_1)11$	$100X(r)X1$	$101X(r)XX0$	$10(f)1$
$11(d_2)X01$	$100XX(r)1$	$101XX(r)X0$	$1(f)01$
$11X(d_3)01$	$100XX1(r)\sqcup$	$101XXX(r)0$	$(f)101$
$11XX(d_3)1$	$100XX(d_1)1$	$101XXX0(r)\sqcup$	$(f)\sqcup 101.$

Aufgabe 2 (*Abgeschlossenheit Entscheidbarer Sprachen,* $2+2+1+1+1+1$ *Punkte*)

Beweisen Sie die folgenden Aussagen. Eine textuelle Beschreibung der Turingmaschinen genügt.

a) Die Menge der entscheidbaren Sprachen ist unter dem Kleen'schen Abschluss abgeschlossen, d.h. für jede entscheidbare Sprache gilt, dass L^* auch entscheidbar ist (entscheidbar $\equiv$ turing-entscheidbar).

b) Die Menge der entscheidbaren Sprachen ist bzgl. der Operation $\min(\cdot)$ abgeschlossen, d.h. für jede entscheidbare Sprache L gilt $\min(L)$ ist entscheidbar.

$\min(L) := \{x \in L \mid \text{kein echtes Präfix von } x \text{ liegt in } L\}$

[Ein Präfix p von x heißt echt, wenn $p \neq x$ bzw. $|p| < |x|$]

Begründen oder widerlegen Sie die folgenden Aussagen (kurze Begründung genügt).

c) Jede unentscheidbare Sprache enthält eine entscheidbare Teilmenge.

d) Jede Teilmenge einer entscheidbaren Sprache ist entscheidbar.

e) Für jede unentscheibare Sprache L gibt es eine echte Obermenge, die ebenfalls unentscheidbar ist.

f) Aus L_1 entscheidbar und $L_1 \cap L_2$ entscheidbar folgt L_2 entscheidbar.

Musterlösung:

a) z.Z.: Gegeben einer Turingmaschine M, welche die Sprache L entscheidet lässt sich eine Turingmaschine M' konstruieren, die L^* entscheidet.

Gegeben einer Eingabe w, teilt M' die Eingabe nichtdeterministisch in Teilworte $w = w_1 \cdot w_2 ... \cdot w_k$. Hierzu wird der Anfang und das Ende jedes Teilworts markiert. Danach verwendet man die

Turingmaschine M um für jedes einzelne Teilwort w_i zu testen ob es in L liegt. Ist dies der Fall, so kann das zusammengesetzte Wort akzeptiert werden.

Um die Teilworte zu testen ohne die ursprüngliche Eingabe zu verändern verwenden wir ein erweitertes Band-Alphabet ($\Gamma' = \Sigma \times \Gamma \times \{b, e, -\}$ = Eingabe-Alphabet × altes Band-Alphabet × Markierungssymbole). Dies ist wichtig, um wärend der Überprüfung eines Teilworts nicht benachbarte Teilworte zu verändern. Vor jedem Test eines Teilworts kann das Band zurückgesetzt werden.

Durch den Nichtdeterminismus beim Aufteilen des Wortes akzeptiert die Maschine genau dann, wenn sich das eingegebene Wort korrekt aufspalten lässt (es genügt wenn zumindest ein Durchlauf akzeptiert). Andere Aufspaltungen führen zu Durchläufen, die nicht akzeptieren.

Korrekturtipp: Für die Korrektur ist ausschlaggebend: 1. Das Wort wurde korrekt aufgeteilt 2. Jedes Teilwort wird einzeln überprüft, und 3. es muss kurz erwähnt werden, dass die Überprüfung durchgeführt werden kann, ohne das Band großartig zu verändern.

b) z.Z.: Gegeben einer Turingmaschine M welche die Sprache L entscheidet, lässt sich eine Turingmaschine M' konstruieren, die $\min(L)$ entscheidet.

Beobachtungen:

- Jedes Wort w mit $|w| = n$ Symbolen hat n echte Präfixe (endlich viele).
- Jede Sprache, die von einer nichtdeterministischen Turingmaschine erkannt werden kann, kann auch von einer deterministischen Turingmaschine erkannt werden (nur mehr Platz und Zeitaufwand). Deshalb sei oBdA M deterministisch.

Zuerst überprüft M', ob das ganze Wort in der Sprache liegt. Danach testen wir für jedes echte Präfix, ob dieses in der Sprache liegt. Dies verläuft ähnlich zu Aufgabenteil a) für jedes Präfix einzeln. Sobald ein echtes Präfix in L liegt, kann das Verfahren abgebrochen werden (nichtakzeptierend). Es ist wichtig, dass M deterministisch ist, weil eine Ausführung einer nichtdeterministischen Turingmaschine nicht genügt um auszuschließen, dass ein Prafix in der Sprache liegt.

Korrekturtipp: Ausschlaggebend sind: 1. Jedes Präfix muss deterministisch getestet werden 2. Die überprüfung darf nicht nichtdeterministisch sein.

c) Die Aussage stimmt, $\emptyset$ ist berechenbar und Teilmenge jeder Sprache. Alternativ: Unentscheidbare Sprachen sind nicht leer ($\emptyset$ ist berechenbar). Mithilfe des Auswahlaxioms lässt sich ein Element aus der Sprache auswählen (oder mehrere); dann ist die (Teil-)Sprache aller ausgewählten Elemente endlich und somit entscheidbar.

d) Diese Aussage ist falsch! Jede Sprache über dem Alphabet Σ ist Teilmenge von Σ^*. Wäre die Aussage korrekt, dann gäbe es keine unentscheidbaren Sprachen.

e) Diese Aussage ist korrekt! Für jedes Wort $w \notin L$ ist die Sprache $L \cup \{w\}$ eine echte Obermenge von L. Wenn L unentscheidbar ist, so ist auch $L \cup \{w\}$ unentscheidbar. Ein solches w ($\notin L$) existiert immer, da $L \neq \Sigma^*$ (Auswahlaxiom).

f) Diese Aussage ist falsch! Es ist zum Beispiel möglich, dass L_1 endlich ist, daraus folgt, unabhängig von L_2,dass $L_1 \cap L_2$ endlich und somit berechenbar ist.

Aufgabe 3 (*2-Kopf-Turingmaschinen,* $4 + 1$ *Punkte*)

Eine 2-Kopf-Turingmaschine besitzt zwei Leseköpfe, die sich unabhängig voneinander bewegen können. Beide Köpfe starten auf dem linkesten Symbol der Eingabe. Die Köpfe agieren synchron, und die Turingmaschine sieht in jedem Schritt beide eingelesenen Zeichen. Das Verhalten von Kopf 1 kann vom Zeichen abhängen, welches Kopf 2 liest.

Die Turingmaschine hat die Form $M_2 = (Q, \Sigma, \Gamma, \delta, S_{\in Q}, F_{\subseteq Q})$. Die Zustandsübergangsfunktion hat die folgende Form (wir betrachten nur deterministische 2-Kopf-Turingmaschinen):

$Q \times \Gamma \times \Gamma \ \rightarrow \ Q \times \Gamma \times \{R, L, N\} \times \Gamma \times \{R, L, N\}$

Für einen Übergang $(q, \gamma_1, \gamma_2) \rightarrow (q', \gamma_1', R, \gamma_2', L)$ sagen wir, wenn die 2-Kopf-Turingmaschine in Zustand q ist, mit Kopf 1 das Zeichen γ_1 liest, und mit Kopf 2 das Zeichen γ_2 liest, dann geht die Maschine über in Zustand q', schreibt mit Kopf 1 γ_1' und bewegt ihn nach rechts, und schreibt mit Kopf 2 γ_2' und geht nach links.

a) Zeigen Sie mithilfe einer Beweisskizze, dass 2-Kopf-Turingmaschinen genauso mächtig sind wie normale (1-Kopf-)Turingmaschinen.

b) Wie viel langsamer als eine 2-Kopf-Turingmaschine kann eine normale Turingmaschine schlimmstenfalls sein? Verwenden Sie Ihre Konstruktion aus Aufgabe a).

Musterlösung:

a) z.Z. Gleich mächtig, deshalb müssen beide Richtungen gezeigt werden $X = Y \Leftrightarrow \ X \supseteq Y \wedge X \subseteq Y$

1. Als erstes müssen wir zeigen, dass 2-Kopf-Turingmaschinen jede Sprache erkennen können, die normale Turingmaschinen erkennen können.

Dies gilt offensichtlich, da man einen der beiden Köpfe vollständig ignorieren kann. Sei $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{R, L, N\}$ die Zustandsübergangsfunktion einer deterministischen Turingmaschine, dann ist $\delta'(q, \gamma_1, \gamma_2) = (q', \gamma_1', d_1, \gamma_2, N)$ gdw. $\delta(q, \gamma_1) =', \gamma_1', d_1)$ eine Zustandsübergangsfunktion einer 2-Kopf-Turingmaschine, die offensichtlich die selbe Sprache erkennt.

2. Wir müssen zeigen, dass normale Turingmaschinen die selben Sprachen erkennen können, wie 2-Kopf-Turingmaschinen.

Gegeben einer deterministischen 2-Kopf-Turingmaschine M_2 wollen wir eine normale Turingmaschine entwerfen.

$M_2 = (Q, \Sigma, \Gamma, \delta, S, F) \qquad \delta : Q \times \Gamma \times \Gamma \rightarrow Q \times \Gamma \times \{R, L, N\} \times \Gamma \times \{R, L, N\}$

Idee: Wir speichern die Position der zwei Köpfe auf dem Band! $\Rightarrow \Gamma' = \Gamma \times \{-, 1\} \times \{-, 2\}$

Um einen Schritt der 2-Kopf-Turingmaschine zu simulieren, muss der Lesekopf der Turingmaschine beide Markierungen besuchen. Betrachte den Schritt $\delta(q, \gamma_1, \gamma_2) = (q', \gamma_1', d_1, \gamma_2', d_2)$

- Jeder Schritt beginnt auf der Bandposition von Kopf 1.
- Die Turingmaschine merkt sich das gelesene Zeichen γ_1 (im Zustand).
- Mit dem Zeichen im Zustand, läuft die Turingmaschine zu der Markierung von Kopf 2 (man kann sich leicht merken in welche Richtung (links oder rechts) die Markierung liegt).
- Dort liest sie das Zeichen γ_2, nun kann $\delta(q, \gamma_1, \gamma_2)$ berechnet werden.
- Die Turingmaschine schreibt γ_2' aufs Band, und bewegt Markierung 2 in Richtung d_2.
- Nun kehrt der Lesekopf zurück zu Markierung 1.
- Dort schreibt er γ_1 und bewegt Markierung 1 in Richtung d_1.
- Nun befindet sich der Kopf erneut über Markierung 1 und der nächste Schritt kann simuliert werden.

b) In dieser Konstruktion benötigt jeder Schritt der Simulation O(Distanz) Schritte. Die Distanz wächst maximal linear mit der Anzahl der Ausführungsschritte n der 2-Kopf-Turingmaschine (beide Köpfe starten auf dem selben Feld und bewegen sich pro Schritt nur um maximal ein Feld). Dementsprechend liegt die Gesamtlaufzeit bei $O(n^2)$.