Theorie der Informatik

G. Röger
Frühjahrssemester 2020

Universität Basel
Fachbereich Informatik

Übungsblatt 9 — Lösungen

Aufgabe 9.1 (Turing-Maschinen; 2 Punkte)

Wir haben das Band einer Turing-Maschine so definiert, dass es in beide Richtungen unendlich ist. Eine alternative Definition verwendet ein Band, dass nur in eine Richtung unendlich ist. Formal gesehen, kann man das erreichen, indem die Definition eines *Berechnungsschritts* ändert und alle anderen Definitionen unverändert lässt: In der Definition auf Folie C7.16, ändern wir den dritten Fall von

$$\langle \varepsilon, q, b_1 \dots b_n \rangle \vdash_M \langle \varepsilon, q', \Box c b_2 \dots b_n \rangle \qquad \text{if } \langle q', c, L \rangle \in \delta(q, b_1), n \geq 1$$

zu

$$\langle \varepsilon, q, b_1 \dots b_n \rangle \vdash_M \langle \varepsilon, q', c b_2 \dots b_n \rangle \qquad \text{if } \langle q', c, L \rangle \in \delta(q, b_1), n \geq 1.$$

Turing-Maschinen mit solchen Berechnungsschritten verhalten sich genau so wie unsere Turing-Maschinen, bis auf den Fall, in dem sie versuchen den Kopf von der ersten Position aus nach links zu verschieben. In diesem Fall bleibt der Kopf einfach auf der ersten Position.
Die Verwendung von beidseitig unendlichem Band macht unsere Turing-Maschinen nicht ausdrucksstärker als Maschinen mit einseitig unendlichem Band. Erklären Sie wie eine gegebene Turing-Maschine $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \Box, E \rangle$ mit beidseitig unendlichem Band zu einer Maschine M' mit einseitig unendlichem Band verändert werden kann, damit M' die gleiche Sprache wie M akzeptiert.
Hinweis: Die Beweisidee ist hier ausreichend, aber Sie sollten erklären, welche Art von zusätzlichen Symbolen und Zuständen in M' benötigt werden und was die grundsätzliche Idee der Transformation ist.

Lösung:

Es gibt verschiedene Arten der Transformation, von denen wir hier zwei vorstellen. Bei der einen wird der gesamte Bandinhalt um eine Position nach rechts verschoben, wenn der Kopf über das linke Ende laufen würde. Bei der anderen Variante, „schneidet“ man das Band links neben der Eingabe in zwei einseitig begrenzte Bänder, die man dann ineinander falten, also immer abwechselnd ein Feld vom linken Band (das wir von rechts nach links repräsentieren) und eines vom rechten Band.
Variante 1:

Die Hauptidee der Transformation ist, den gesamten Bandinhalt immer einen Schritt nach rechts zu verschieben, wenn der Lesekopf an der ersten Position steht. Das technische Problem damit ist, dass wir $\Box$ nicht dazu verwenden können das Ende des Bandinhalts zu erkennen. (Die Maschine M könnte ja ein $\Box$ in die Mitte des Wortes schreiben.) Wir verwenden also zwei neue Symbole $\boxed{S}$ und $\boxed{E}$ für des Start und das Ende des aktuellen Bandinhalts.
Am Anfang von M' ersetzen wir das erste Symbol durch ein $\boxed{S}$ und bewegen den Kopf um eins nach rechts. Danach ersetzen wir wiederholt das Symbol unter dem Lesekopf mit dem zuletzt überschriebenen Symbol. Damit das möglich ist, brauchen wir einen neuen Zustand q_x pro Bandsymbol $x \in \Sigma$. Wenn wir in Zustand q_x ein y lesen, ersetzen wir es durch ein x und gehen in Zustand q_y (so wird das nächste Zeichen durch ein y ersetzt). Auf diese Art fahren wir fort, bis wir das erste Blank lesen und hängen ein $\boxed{E}$ an. Danach geht M' nach links bis zum Start der Eingabe (eine Position rechts von $\boxed{S}$). Mit dieser letzten Bewegung wechseln wir in den Startzustand von M. Anstelle der ursprünglichen Konfiguration $\langle \varepsilon, q_0, w \rangle$ sind wir jetzt in der Konfiguration $\langle \boxed{S}, q_0, w\boxed{E} \rangle$.
Die Maschine M' verwendet jetzt die gleichen Zustände wie M, allerdings erweitern wir die Übergangsfunktion, so dass jeder Zustand zwei zusätzliche ausgehende Kanten für $\boxed{S}$ und $\boxed{E}$ hat. Wenn ein $\boxed{E}$ gelesen wird, wird es durch ein $\Box$ ersetzt und ein Feld weiter rechts wird ein

neues $\boxed{E}$ geschrieben. Danach geht M' zurück in den Zustand, dem sie vorher war. Da wir uns den Zustand nicht „speichern“ können, müssen wir für dieses Verschieben einen neuen Zustand pro Zustand von M einführen. Nach dieser Transformation ist M' im gleichen Zustand wie vorher und zeigt jetzt auf das neu eingefügte $\square$ genau so wie M das in dieser Situation tun würde. Wir sind damit von der Konfiguration $\langle \boxed{S}\alpha, q, \beta\boxed{E}\rangle$ zur Konfiguration $\langle \boxed{S}\alpha, q, \beta\square\boxed{E}\rangle$ gekommen.
Wenn statt dessen in einem Zustand ein $\boxed{S}$ gelesen wird, müssen wir die gesamte Eingabe um eine Position nach rechts verschieben. Das Symbol $\boxed{S}$ bleibt an erster Position, das zweite Zeichen wird mit einem $\square$ ersetzt, das dritte Zeichen wird mit dem zweiten ersetzt, usw. Dieses Verschieben funktioniert genau wie das Verscheiben am Anfang von M', nur das wir dieses Mal auch $\square$ mit verschieben und so lange weitermachen, bis wir $\boxed{E}$ verschoben haben. Danach bewegen wir den Kopf zurück nach links bis wir $\boxed{S}$ finden und von da eine Position nach rechts. In diesem letzten Schritt gehen wir wieder in den Zustand, in dem wir das Verschieben ausgelöst haben. Wie oben gesagt, brauchen wir die Struktur zum Ersetzen einmal für jeden Zustand von M, damit wir uns merken können, in welchen Zustand wir zurückwechseln müssen. Wir kommen damit von der Konfiguration $\langle \varepsilon, q, \boxed{S}\beta\boxed{E}\rangle$ zur Konfiguration $\langle \boxed{S}, q, \square\beta\boxed{E}\rangle$.
Mit diesen Erweiterungen verhält sich M' genau wie M, nur dass zwischendurch die Verarbeitung unterbrochen wird um die Eingabe auf dem Band zu verschieben und neue Blanks einzufügen.

Variante 2:

Für die Erläuterung der Idee benennen wir die unendlich vielen Felder, so dass p_0 die erste Position der Eingabe bezeichnet und $p_1, p_2, \ldots$ die Felder rechts daneben bezeichnen. Links von p_0 befinden sich (von rechts nach links) die Felder $p_{-1}, p_{-2}, \ldots$.
Die Idee ist, die Maschine mit einer linksseitig beschränkten TM zu simulieren, deren Band den Inhalt der ursprünglichen Felder in der Reihenfolge $p_0, p_{-1}, p_1, p_{-2}, p_2, \ldots$ repräsentieren. Die Kernidee ist dabei, dass der Kopf bei einer Linksbewegung bei positiven Index immer zwei Positionen nach links geht, bei einer Rechtsbewegung bei nicht-negativem Index zwei nach rechts, etc. Bei den negativen Index benötigen wir zusätzlich eine Umkehrung der Richtung. Natürlich existieren die Indizes nicht wirklich, so dass wir das in das Programm der TM aufnehmen müssen.
Hierbei verwenden wir zunächst für jeden Zustand q der originalen TM sechs Zustände $q_+, q_-, q_+^R, q_-^R, q_+^L$ und q_-^L. Das Subskript kennzeichnet, ob wir uns gerade auf den positiven oder den negativen Positionen bewegen. Die Zustände mit Superskript R bzw. L dienen als Zwischenzustand, um zu kennzeichnen, dass ein zweiter Schritt in die angegebene Richtung notwendig ist, der dann zu dem entsprechenden Zustand ohne Superskript führen soll.
Hierbei haben wir jedoch noch nicht den Wechsel zwischen positiven und negativen Positionen bedacht. Um diesen korrekt hinzukriegen, schieben wir zu Beginn den gesamten Bandinhalt eine Position nach rechts und markieren das linke Ende mit einem neuen Symbol # (unter Zuhilfenahme zusätzlicher Zustände, vgl. Variante 1).
Sind wir nun in einem Zustand q_+^L (also ausgehend von einer nicht-negativen Position auf dem Weg nach links, aber erst einen Schritt weit gegangen), gehen wir nur dann einen weiteren Schritt nach links und in Zustand q_+, wenn wir nicht # lesen. Andernfalls sind wir von Position 0 losgelaufen und wollen auf Position -1, die zwei Positionen rechts der aktuellen Zelle zu finden ist. Die TM bewegt den Kopf also mit Hilfe eines weitern Zustands q_0^R dorthin und geht dann in Zustand q_-.
Der Wechsel von den negativen Positionen auf die Positiven funktioniert ähnlich, allerdings löst eine ursprüngliche Rechtsbewegung weiterhin erst zwei Linksbewegungen aus und erst dann wird getestet, ob der Kopf wieder einen Schritt nach rechts auf die nichtnegativen Positionen bewegt werden muss.

Aufgabe 9.2 (Multiplikation ist Turing-berechenbar; 2 Punkte)

Beschreiben Sie die Beweisidee dafür, dass die Multiplikation von Binärzahlen Turing-berechenbar ist, d.h., beschreiben Sie eine Turing-Maschine, die mul^{code} für die Funktion $mul : \mathbb{N}_0^2 \to_{\text{p}} \mathbb{N}_0$ mit $mul(n_1, n_2) = n_1 \cdot n_2$ berechnet.
Hinweis: Sie können verwenden, dass die Addition Turing-berechenbar ist, und eine high-level Beschreibung der Turing-Maschine ist ausreichend.

Lösung:

Die Hauptidee hier ist, eine Zahl mit 0 zu initialiseren und dann n_1-mal die Zahl n_2 zu addieren. Um zu zählen, wie oft noch addiert werden muss, können wir nach jeder Iteration n_1 um 1 reduzieren und stoppen, wenn n_1 dadurch 0 wird. Die einzelnen Bestandteile sind alle aus der Vorlesung bekannt (Addition von zwei Zahlen, Subtraktion von 1) oder sehr einfach umzusetzen (Test auf 0, einen Bereich auf dem Band auf 0 initialisieren).

Das Problem ist, dass die Berechnung einer Turing-Maschine ihre Eingabe konsumiert und auch in Bereiche ausserhalb der Eingabe schreiben kann. So stehen zum Beispiel nach der Addition die beiden Eingabezahlen nicht mehr auf dem Band. Wir müssen daher verschiedene Bereiche auf dem Band voneinander trennen. Jede Maschine für eine Teiloperation wird auf „ihren"Bereich des Bands beschränkt. Wenn ein Bandbereich nicht ausreicht, kann er durch Verschieben anderer Bereiche vergrössert werden. Um die verschiedenen Operationen hintereinander auszuführen, muss unsere Maschine also nur die richtigen Eingaben auf den entsprechenden Bandbereich schreiben und in den Intialzustand der spezialisierten Maschine (zum Beispiel der für Addition) übergehen. Anstatt in einen akzeptierenden Zustand überzugehen, wird diese Maschine dann so verändert, dass sie in den nächsten Schritt unserer Maschine übergeht. In diesem Fall sollte zum Beispiel nach der Addition die Zahl n_1 um 1 reduziert werden.

Aufgabe 9.3 (Komposition berechenbarer Funktionen ist berechenbar; 2 Punkte)

Seien $f : \Sigma^* \to \Sigma^*$ und $g : \Sigma^* \to \Sigma^*$ Turing-berechenbare partielle Funktionen für ein Alphabet Σ. Zeigen Sie, dass dann auch die *Komposition* $(f \circ g) : \Sigma^* \to \Sigma^*$ Turing-berechenbar ist.

Die Komposition zweier Funktionen ist allgemein definiert als $(f \circ g)(x) = f(g(x))$. Der Funktionswert ist insbesondere auch undefiniert, wenn $g(x)$ undefiniert ist.

Lösung:

Wesentliche Ideen:

Wenn f und g Turing-berechenbar sind, dann gibt es DTMs M_f und M_g, die auf Eingabe y das Wort $f(y)$ bzw. auf Eingabe x das Wort $g(x)$ berechnen.

Wenn wir die DTM M_g auf einer Eingabe x starten, für die $g(x)$ definiert ist, dann sind nach Definition der von einer DTM berechneten Funktion Bandinhalt und Kopfposition beim Terminieren in genau der Form, in der sie auch für die Eingabe von M_f benötigt werden. Daher reicht es in diesem Fall aus, M_f und M_g zu einer gemeinsamen DTM zu kombinieren, indem alle Übergänge auf Endzustände von M_g stattdessen auf den Startzustand von M_f führen. Damit berechnet die Maschine zunächst $g(x)$ auf der Eingabe x. Dies ist dann die Eingabe y für f, so dass insgesamt $f(y) = f(g(x)) = (f \circ g)(x)$ berechnet wird.

Technische Details:

Wir gehen ohne Einschränkung davon aus, dass die Zustandsmengen von M_f und M_g disjunkt sind (sonst Zustände umbenennen) und ihre Bandalphabete identisch sind (sonst durch die Vereinigung der beiden Bandalphabete ersetzen; die dadurch neu zu definierenden Transitionen, z.B. wenn M_g ein Zeichen liest, das nur im Bandalphabet von M_f vorkommt, können beliebig gewählt werden, da sie nie benötigt werden).

Ein technisches Problem entsteht hierbei allerdings, wenn M_g in einer Konfiguration terminiert, die keine gültige Berechnung repräsentiert (weil der Kopf in der falschen Position steht oder unerlaubte Zeichen auf dem Band stehen). In diesem Fall sollte die kombinierte Maschine kein gültiges Ergebnis liefern, da $(f \circ g)(x)$ undefiniert sein soll, aber ohne weitere Forderungen an M_f können wir das nicht garantieren.

Um dieses Problem zu beheben, ist es am einfachsten, die Maschine M_g so zu modifizieren, dass sie nie in einer ungültigen Konfiguration anhält, sondern stattdessen in eine Endlosschleife geht. Dies ist zum Beispiel zu erreichen, indem man am Ende der Berechnung von M_g den Bandinhalt auf korrekte Form prüft. Eine Schwierigkeit ist hierbei, festzustellen, wo der besuchte Teil des Bandes beginnt und endet. Dies kann man am einfachsten lösen, indem man während der Berechnung von M_g nie das Zeichen $\square$ sondern stattdessen ein neues Zeichen $\hat{\square}$ schreibt, das sich ansonsten wie $\square$ verhält. Dann ist sicher gestellt, dass bei der Prüfung des Ergebnisses das Zeichen $\square$ auf beiden

Seiten das Ende des besuchten Bandes markiert. Bei der abschliessenden Prüfung kann dann $\hat{\square}$ wieder durch $\square$ ersetzt werden.

Aufgabe 9.4 (Aufzählungsfunktionen; 2 Punkte)
Sei $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. Geben Sie totale und berechenbare Funktionen $f : \mathbb{N}_0 \to \Sigma^*$ an, die die folgenden Sprachen rekursiv aufzählen, und geben Sie jeweils die Funktionswerte $f(0)$, $f(1)$, ..., $f(5)$ an. Sie dürfen hierbei alle in der Vorlesung besprochenen berechenbaren Funktionen verwenden.

(a) $L_1 = \{\mathtt{b}^n\mathtt{a}^{2n} \mid n \in \mathbb{N}_0, n \text{ ist gerade}\}$

Lösung:

$$f_{L_1}(x) = \begin{cases} \mathtt{b}^x\mathtt{a}^{2x} & \text{falls } x \text{ gerade} \\ \varepsilon & \text{sonst} \end{cases}$$

x	0	1	2	3	4	5
$f_{L_1}(x)$	ε	ε	bbaaaa	ε	bbbbaaaaaaaa	ε

(b) $L_2 = \{w \in \Sigma^* \mid \mathtt{a} \text{ kommt in } w \text{ genau einmal vor}\}$

Lösung:

$f_{L_2}(x) = \mathtt{b}^{decode_1(x)}\mathtt{ab}^{decode_2(x)}$

x	0	1	2	3	4	5
$decode_1(x)$	0	0	1	0	1	2
$decode_2(x)$	0	1	0	2	1	0
$f_{L_2}(x)$	a	ab	ba	abb	bab	bba

Aufgabe 9.5 (Entscheidbarkeit; 2 Punkte)
Welche der folgenden Aussagen sind korrekt? Begründen Sie Ihre Antworten mit jeweils ein bis zwei Sätzen.

(a) Wenn L entscheidbar ist, dass ist L auch endlich.

Lösung:

Die Aussage stimmt nicht. Zum Beispiel ist the Sprache aller Wörter, die mit a anfangen unendlich, aber entscheidbar.

(b) Wenn L regulär ist, dann ist L auch entscheidbar.

Lösung:

Die Aussage stimmt. Für jede reguläre Sprache gibt es einen DFA, diesen kann man in endlicher Zeit auf einer Eingabe simulieren um zu entscheiden, ob das Wort zur Sprache gehört, oder nicht.

(c) Wenn L_1 und L_2 entscheidbar sind, dann ist $L_1 \cap L_2$ entscheidbar.

Lösung:

Die Aussage stimmt. Da L_1 und L_2 entscheidbar sind, existieren Turing-Maschinen, die deren charakteristische Funktion berechnen. Da diese Maschinen immer halten, können wir sie nacheinander ausführen und 1 ausgeben, wenn beide 1 ausgeben.

(d) Wenn L kontext-sensitiv ist, dann ist L auch semi-entscheidbar.

Lösung:

Die Aussage stimmt. Jede kontext-sensitive Sprache ist auch eine Typ-0-Sprache und wir haben gezeigt, dass diese genau die semi-entscheidbaren Sprachen sind.