Prof. T. Nipkow
K. Kappelmann, J. Rädle, L. Stevens **ABGABE 15.06.2020, 23:59 (Σ 4,5 P.)** Technische Universität München
Lehrstuhl für Logik und Verifikation

Einführung in die Theoretische Informatik

Sommersemester 2020 – Übungsblatt Lösungsskizze 7

AUFGABE 7.1. (*Here we convert again*)

Konstruieren Sie eine Turingmaschine $T = (Q, \{|\}, \{|, 0, 1\}, \delta, q_0, \square, F)$, die als Eingabe eine mit | unär kodierte Zahl nimmt. Die gegebene Zahl soll in die entsprechende Binärzahl im Little-Endian-Format umgewandelt werden.. Beispielsweise soll T die Strichfolge |||| durch die Folge 001 ersetzen und dann in einem Endzustand halten. Die leere Eingabe kodiert 0. Ihre Lösung soll die Konvention aus der Vorlesung, dass Endzustände keine ausgehenden Kanten haben dürfen, respektieren. Es wird eine Webseite bereitgestellt, auf der Sie die Turingmaschine konstruieren, visualisieren und mit Eingaben testen können. Schauen Sie sich dort insbesondere das Beispiel *binary increment* an. Die Abgabe der Aufgabe erfolgt mit TUMJudge, laden Sie sich das Template auf Moodle herunter in dem erklärt wird wie die Abgabe aussehen soll. 1P

Lösungsskizze

Lösungsidee: wir schreiben zunächst eine 0 rechts hinter den letzten Strich und initialisieren damit die auszugebende Binärzahl. Dann verwenden wir die Maschine aus dem Beispiel *binary increment*, um Binärzahl so oft zu inkrementieren, wie anfangs Striche auf dem Band stehen. Die Turingmaschine im Format der oben genannten Webseite schaut folgendermaßen aus:

```
input: ''
blank: ' '
final states: final
start state: init
table:
  start:
    '|': {write: ' ', R: scanR}
    ['0','1']: {N: final}
  init:
    '|': R
    ' ': {write: 0, L: reset}

  scanR:
    '|': R
    [0,1,' ']: {N: carry}

  reset:
    [0,1,'|']: L
    ' ': {R: start}

  carry:
    1: {write: 0, R}
    0: {write: 1, N: reset}
    ' ': {write: 1, N: reset}

  final:
```

AUFGABE 7.2. (*. . . sehen deine Schuhe prima aus!*) 2P

Wir definieren eine alternative Akzeptanzbedingung für Schleifomaten: Ein Schleifomat A mit Endzuständen ist ein 7-Tupel $(Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$. Die ersten 6 Einträge bleiben unverändert wie auf Hausaufgabenblatt 6 definiert. Der neue Eintrag $F \subseteq Q$ kennzeichnet die Endzustände des Schleifomats. Die mit Endzuständen akzeptierte Sprache eines Schleifomats ist definiert als

$$L_F(M) \coloneqq \left\{ w \in \Sigma^* \mid \exists q_f \in F, \gamma \in \Gamma^*.\ (q_0, w, Z_0) \to_M^* (q_f, \varepsilon, \gamma) \right\},$$

wobei γ die Ausgabe der Berechnung des Schleifomats ist. Wie auch bei Turingmaschinen nehmen wir an, dass $\delta(q_f, a)$ nicht definiert ist für $q_f \in F$ und $a \in \Gamma$.

Zeigen Sie nun, dass jede Turingmaschine durch einen Schleifomat mit Endzuständen simuliert werden kann, indem Sie eine passende Konstruktion formal angeben (d.h. jede Komponente des 7-Tupels muss präzise definiert sein). Dabei soll sowohl die Akzeptanz eines Wortes als auch die Ausgabe der Turingmaschine simuliert werden. Letzteres bedeutet, dass falls eine TM mit Band $\square \cdots \square w_1 \cdots w_n \square \cdots \square$ mit $w_1, \ldots, w_n \in \Gamma$ hält, die Schleife

schlussendlich bei $\square^k w_1 \cdots w_n \square^m$ für beliebiges $k, m \in \mathbb{N}$ stehen soll. Ein ungültiger, finaler Schleifeninhalt wäre hingegen $w_2 \cdots w_n w_1$. Die finale Kopfposition der TM muss nicht ersichtlich sein.
Beschreiben Sie dabei auch die Idee ihrer Konstruktion informell (ansonsten erheblicher Punkteabzug)! Die Korrektheit der Konstruktion müssen Sie nicht beweisen.

Lösungsskizze
Sei $M = (Q, \Sigma, \Gamma, q_0, \delta, \square, F)$ eine deterministische Turingmaschine, die wir durch einen Schleifomat 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 Schleife ablegen. Der Schleifomat kann die Schleife rotieren, indem er ein Schleifenzeichen liest, und danach wieder ans Ende der Schleife anhängt. So lässt sich die Schleife flexibel manipulieren. Für jeden Schritt der Turingmaschine muss allerdings die Schleife einmal komplett durchlaufen werden. Wenn ein Endzustand erreicht wird, wird der Zustand und das #-Symbol von der Schleife gelöscht und der Schleifomat akzeptiert.
Unser Schleifomat $A \coloneqq \big(Q', \Sigma, \Gamma \cup Q \cup \{\#\}, \mathit{init}, \#, \delta', F\big)$ hat die Zustandsmenge

$$Q' \coloneqq \{\mathit{init}, \mathit{copy}, \mathit{start}\} \cup \{\mathit{find}_x \mid x \in \Gamma\} \cup \{\mathit{step}_x^q \mid x \in \Gamma, q \in Q\} \cup \{\mathit{ff}^q, \mathit{clean}^q \mid q \in Q\} \cup F$$

und die Übergänge

$\delta'(\mathit{start}, \varepsilon, x)$	$\to$	$(\mathit{find}_x, \varepsilon)$	$\forall x \in \Gamma$	Zum Kopf vorlaufen, dabei
$\delta'(\mathit{find}_x, \varepsilon, y)$	$\to$	(find_y, x)	$\forall x, y \in \Gamma$	jeweils letztes Zeichen merken.
$\delta'(\mathit{find}_x, \varepsilon, q)$	$\to$	$(\mathit{step}_x^q, \varepsilon)$	$\forall x \in \Gamma, q \in Q$	Zustand auch merken.
$\delta'(\mathit{step}_x^q, \varepsilon, y)$	$\to$	$\begin{cases} (\mathit{ff}^q, xq'z), & \text{falls } \delta(q,y) = (q', z, N) \\ (\mathit{ff}^q, xzq'), & \text{falls } \delta(q,y) = (q', z, R) \\ (\mathit{ff}^q, q'xz), & \text{falls } \delta(q,y) = (q', z, L) \end{cases}$	$\forall x, y \in \Gamma, q \in Q$	Schritt ausführen
$\delta'(\mathit{ff}^q, \varepsilon, x)$	$\to$	(ff^q, x)	$\forall x \in \Gamma, \forall q \in Q$	Vorspulen bis zum #...
$\delta'(\mathit{ff}^q, \varepsilon, \#)$	$\to$	$(\mathit{start}, \#)$	$\forall q \in Q \setminus F$	... und von vorne.
$\delta'(\mathit{ff}^q, \varepsilon, \#)$	$\to$	$(\mathit{clean}^q, \#)$	$\forall q \in F$	Zurück zum Start, um den Zustand zu löschen.
$\delta'(\mathit{clean}^q, \varepsilon, x)$	$\to$	(clean^q, x)	$\forall x \in \Gamma, \forall q \in F$	Vorspulen zum Endzustand.
$\delta'(\mathit{clean}^q, \varepsilon, q)$	$\to$	$(\mathit{clean}^q, \varepsilon)$	$\forall q \in F$	Zustand löschen
$\delta'(\mathit{clean}^q, \varepsilon, \#)$	$\to$	(q, ε)	$\forall q \in F$	Lösche # und akzeptiere
$\delta'(\mathit{start}, \varepsilon, q)$	$\to$	$(\mathit{step}_\square^q, \varepsilon)$	$\forall q \in Q$	Sonderfälle für Rand
$\delta'(\mathit{step}_x^q, \varepsilon, \#)$	$\to$	$(\mathit{start}, xq\square\#)$	$\forall x \in \Gamma, q \in Q$	

Nun gilt (hier ohne Beweis):

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

Um dafür zu sorgen, dass auch wirklich $L(A) = L(M)$ gilt, muss man den Schleifomat noch so erweitern, dass er am Anfang die initiale Konfiguration in die Schleife schreibt:

$\delta'(\mathit{init}, x, \#) \to (\mathit{copy}, q_0 x\#)$	$\forall x \in \Sigma$	Initialzustand und erstes Zeichen schreiben
$\delta'(\mathit{copy}, \varepsilon, x) \to (\mathit{copy}, x)$	$\forall x \in \Sigma \cup \{q_0\}$	Vorspulen bis zum #
$\delta'(\mathit{copy}, x, \#) \to (\mathit{copy}, x\#)$	$\forall x \in \Sigma$	Nächstes Zeichen einlesen
$\delta'(\mathit{copy}, \varepsilon, \#) \to (\mathit{start}, \#)$		Einlesen beendet. Starte Simulation
$\delta'(\mathit{init}, \varepsilon, \#) \to (\mathit{start}, q_0\#)$		Sonderfall für leere Eingabe

AUFGABE 7.3.

1,5P

Entscheiden Sie ob die folgenden Aussagen wahr oder falsch sind. Begründen Sie ihre Behauptung. (Ein ausführlicher formaler Beweis ist nicht gefordert, aber eine umfassende Begründung)

(a) Sei k eine Konstante. Die Klasse der Turingmaschinen die ihren Kopf maximal k mal bewegen können akzeptiert genau die regulären Sprachen. (Transitionen der Form $\delta(q, a) = (q', b, N)$ zählen dabei nicht als Bewegung)
(b) Die Klasse der Turingmaschinen die ihren Kopf nur nach rechts bewegen können (d.h. jede Transition ist von der Form $\delta(q, a) = (q', b, R)$) akzeptiert genau die regulären Sprachen.
(c) Für eine beliebige, aber feste, Haskell Funktion P mit n Argumenten ist die Funktion

$$f_P(x) = \begin{cases} 1 \text{ falls } P \text{ für alle (typkorrekten) Inputs } x_1, \ldots, x_n \text{ terminiert} \\ 0 \text{ sonst} \end{cases}$$

berechenbar.

Lösungsskizze

(a) Falsch. Solche Maschinen können höchstens k Zeichen des Inputs lesen, damit können unendliche Sprachen nicht akzeptiert werden. Ein Beispiel für eine unendliche reguläre Sprache wäre $\{|w|_{\mathsf{b}} = 0 \mid w \in \{\mathsf{a}, \mathsf{b}\}^*\}$.

(b) Wahr. Man kann jeden DFA simulieren indem man eine TM mit den gleichen Zuständen konstruiert, die die Zeichen der Eingabe nacheinander vom Band liest und ihren Zustand entsprechend den Transitionen des DFA ändert. Dabei wird der Kopf nur nach rechts bewegt.
In die andere Richtung kann man jede Transition $\delta(q, a) = (q', b, R)$ einer solchen TM durch eine Transition $\delta(q, a) = q'$ in einem DFA simulieren, da das neu geschriebene Zeichen b der TM nicht mehr gelesen werden kann und daher irrelevant ist.

(c) Wahr. Diese Funktion ist für jede Funktion P entweder konstant 0 oder 1 und daher berechenbar.

AUFGABE 7.4. (***Bonusaufgabe***) 0,5 + 1 Bonuspunkte

Mit dem Verfahren auf Folie 158 der Vorlesung kann man für eine CFG G eine CFG G' in Chomsky-Normalform konstruieren, so dass $L(G') = L(G) \setminus \{\varepsilon\}$ gilt. Auf Folie 152 wird behauptet dass man die Produktion $S \to \varepsilon$ zu G' hinzufügen kann um G'' mit $L(G'') = L(G)$ zu erhalten, wenn $\varepsilon \in L(G)$.

(a) Man füge die Produktion $S \to \varepsilon$ zu einer beliebigen Grammatik G mit Startsymbol S in CNF hinzu um die Grammatik G' zu erhalten. Zeigen Sie dass man daraus im Allgemeinen nicht $\mathsf{L}(G') = \mathsf{L}(G) \cup \{\varepsilon\}$ folgern kann.

(b) Sei G eine beliebige CFG mit Startsymbol S und $\varepsilon \in \mathsf{L}(G)$. Seien G' und G'' die Grammatiken die daraus nach dem Verfahren von oben hervorgehen. Zeigen oder widerlegen Sie $\mathsf{L}(G'') = \mathsf{L}(G)$.

Lösungsskizze

(a) Die Grammatik $S \to AS \mid \mathsf{b} \mid \varepsilon, A \to \mathsf{a}$ erzeugt z.B. das Wort a, das von $S \to AS \mid b, A \to \mathsf{a}$ nicht erzeugt wird.

(b) Sei G eine beliebige Grammatik mit $\varepsilon \in \mathsf{L}(G)$ und G' und G'' die daraus entsprechend der Aufgabenstellung hervorgehenden Grammatiken. Wir bezeichnen die Grammatiken, die nach den Schritten 1-4 auf Folie 158 aus G entstehen, als $G_1 \dots G_4$. (Wobei $G_4 = G'$) In der Vorlesung wurde bereits $\mathsf{L}(G) = \mathsf{L}(G_1) = \mathsf{L}(G_2)$ gezeigt.

Zunächst beobachten wir dass es aufgrund von $\varepsilon \in \mathsf{L}(G) \implies \varepsilon \in \mathsf{L}(G_2)$ eine Ableitung

$$S \to^*_{G_2} X \to_{G_2} \varepsilon$$

geben muss.
Daher wird in Schritt 3 des Algorithmus (Folie 153) die Produktion $S \to \varepsilon$ zu $\hat{P}$ hinzugefügt. $(*)$

Außerdem werden in Schritt 4 (Folie 156) nur Kettenproduktionen abgekürzt, wenn es also eine Produktion $X \to_{G_4} AB$ gibt die nicht in G_3 ist, so muss es eine Ableitung der Form $X \to_{G_3} Y_1 \to_{G_3} \dots \to_{G_3} Y_n \to_{G_3} AB$ geben. $(**)$

Wir zeigen $L(G'') = L(G) = L(G') \cup \{\varepsilon\}$ durch Mengeninklusion in beide Richtungen:

- $\mathsf{L}(G') \cup \{\varepsilon\} \subseteq \mathsf{L}(G'')$: Durch das Hinzufügen von $S \to \varepsilon$ kann ε erzeugt werden. Alle $w \in \mathsf{L}(G')$ können offensichtlich auch von G'' erzeugt werden.
- $\mathsf{L}(G'') \subseteq \mathsf{L}(G') \cup \{\varepsilon\}$:
  Sei $w \in \mathsf{L}(G'')$ mit $w \neq \varepsilon$. Wird die Produktion $S \to \varepsilon$ in dessen Ableitung nicht verwendet gilt offensichtlich $w \in \mathsf{L}(G')$. Wir betrachten also eine Ableitung in der diese Produktion mindestens einmal verwendet wird:

  $$S \to^*_{G''} \alpha S \beta \to_{G''} \alpha\beta \to^*_{G''} w$$

  Da G'' in CNF ist (modulo $S \to \varepsilon$), muss das S in $\alpha S \beta$ durch eine Produktion der Form $X \to_{G''} AS$ oder $X \to_{G''} SA$ eingeführt worden sein. Wir betrachten im Folgenden die erste Form, der andere Fall ist analog. Die Ableitung sieht also aus wie folgt:

  $$S \to^*_{G''} \alpha' X \beta' \to_{G''} \alpha AS \beta \to^*_{G''} w$$

  Die Produktion $X \to_{G''} AS$ ist auch in G_4 enthalten , und ist daher entweder auch direkt in G_3 enthalten oder kann mit $(**)$ in G_3 simuliert werden, in Form einer Ableitung $X \to^*_{G_3} Y \to_{G_3} AS$.
  Wenn die Produktion $X \to_{G_3} AS$ oder $Y \to_{G_3} AS$ bei der Erstellung von G_3 entsteht (oder wenn sie schon in G_2 enthalten war), muss aber wegen $(*)$ auch die Produktion $X \to_{G_3} A$ bzw. $Y \to_{G_3} A$ entstehen.
  Damit gibt es also eine Ableitung $X \to^*_{G_3} A$, womit wir eine Ableitung von w in G_3 ohne Verwendung von $S \to \varepsilon$ bauen können. Schritt 4 verändert die Sprache laut Vorlesung nicht, also können wir w auch in G_4 erzeugen.

> . . . so that one day, every application you run on your computer, can be run within PowerPoint.
>
> — Tom Wildenhain – On The Turing Completeness of PowerPoint