Algorithmen und Datenstrukturen 2

ALGO2 · WiSe-2023/24 · tcs.uni-frankfurt.de/algo2/ · 2024-02-05 · 7ae310c

Übungen zu Woche 9: Turingmaschinen

Dienstag

Moodle 🐣 **9.1 Was wird berechnet.** Gegeben ist die Turingmaschine $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, \texttt{acc}, \texttt{rej}, \square)$ mit $Q = \{q_0, q_1, q_2, q_3, q_4, \texttt{acc}, \texttt{rej}\}$, $\Sigma = \{0, 1\}$ und $\Gamma = \Sigma \cup \{\emptyset, \not{1}, \square\}$. Die Überführungsfunktion δ ist gegeben durch:

$$\begin{aligned}
\delta(q_0, 0) &= (q_1, \emptyset, +1) & \qquad \delta(q_3, \not{1}) &= (q_3, \not{1}, -1)\\
\delta(q_0, \square) &= (\texttt{acc}, \square, +1) & \qquad \delta(q_3, 0) &= (q_3, 0, -1)\\
\delta(q_0, \not{1}) &= (q_4, \not{1}, +1) & \qquad \delta(q_3, \emptyset) &= (q_0, \emptyset, +1)\\
\\
\delta(q_1, 0) &= (q_1, 0, +1) & \qquad \delta(q_4, \not{1}) &= (q_4, \not{1}, +1)\\
\delta(q_1, \not{1}) &= (q_1, \not{1}, +1) & \qquad \delta(q_4, \square) &= (\texttt{acc}, \square, -1)\\
\delta(q_1, 1) &= (q_2, \not{1}, +1) & &\\
\\
\delta(q_2, 1) &= (q_3, \not{1}, -1) & \qquad \delta(q, a) &= (\texttt{rej}, a, +1) \text{ sonst}
\end{aligned}$$

a) 👍 Führe M_1 auf den beiden Eingaben `001111` und `00011111` aus. Gib jeweils die Konfiguration nach Ende der Berechnung, also bei Erreichen von `acc` oder `rej`, an.

b) 🔑 Welche Sprache entscheidet M_1?

Moodle 🐣 🔑 **9.2 Finde den Fehler.** In Abschnitt 6.3 aus Ericksons „Models of Computation“ ist eine Turingmaschine für die Sprache $L = \{0^n 1^n 0^n \mid n \geq 0\}$ angegeben. Diese enthält einige Fehler. Warum entscheidet die Turingmaschine nicht die Sprache L und wie lassen sich die Fehler beheben?

🎂 **9.3 Konstruiere eine Turingmaschine.** Sei $L = \{0^n 1^m 0^{(n+m)} \mid n, m \in \mathbb{N}\}$.

a) 🔑 Beschreibe auf informelle Art und Weise eine Turingmaschine, die die Sprache L entscheidet.

b) 🔭 Gib die beschriebene Turingmaschine formal an. Insbesondere ist also auch die Überführungsfunktion δ der Maschine anzugeben.

Donnerstag

9.4 Rekursive Aufzählbarkeit. Eine Sprache $L \subseteq \Sigma^*$ heißt *rekursiv aufzählbar*, wenn $L = \emptyset$ oder es eine berechenbare, totale Funktion $f : \{0, 1\}^* \to \Sigma^*$ gibt, sodass $f(\{0, 1\}^+) = L$ gilt, das heißt, es gilt

$$L = \{f(x) \mid x \in \{0, 1\}^*\} = \{f(0), f(1), f(00), f(01), f(10), f(11), f(000), \dots\}.$$

a) 🐣 🔑 Zeige, dass jede rekursiv aufzählbare Sprache semi-entscheidbar ist.

b) 🎂 🔭 Zeige, dass jede semi-entscheidbare Sprache rekursiv aufzählbar ist.
Hinweis: Du kannst ohne Beweis verwenden, dass es eine berechenbare bijektive Funktion $t : \Sigma^* \times \Sigma^* \to \Sigma^*$ gibt, deren inverse Funktion ebenfalls berechenbar ist. Das heißt eine Turingmaschine kann für zwei binär kodierte Zahlen $a, b \in \Sigma^*$ eine Zahl $t(a, b)$ berechnen, sodass $t(a, b) \neq t(a', b')$ für alle $(a, b) \neq (a', b')$ gilt, und eine zweite Turingmaschine kann für jedes c das eindeutige Paar (a, b) berechnen, sodass $t(a, b) = c$ gilt. (Dies ist mit der Cantor-Abzählung möglich.)

Weitere Aufgaben und Projekte

Moodle 🔑 **9.5 Turingmaschine simulieren (Coderunner).** Implementiere ein Programm, das eine Turingmaschine als Eingabe erhält, diese simuliert, und die Ausgabe der Turingmaschine berechnet. Die genaue Ein- und Ausgabespezifikation des geforderten Programms findest du in Moodle.

👍 **9.6 Was wird berechnet.** Gegeben ist die Turingmaschine $M_2 = (Q, \Sigma, \Gamma, \delta, q_0, \texttt{halt}, \square)$ mit $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, \texttt{halt}\}$, $\Sigma = \{0, 1\}$ und $\Gamma = \Sigma \cup \{\emptyset, \not{1}, \underline{0}, \underline{1}, \underline{\emptyset}, \underline{\not{1}}, \square\}$. Die Überführungsfunktion δ ist gegeben durch:

$\delta(q_0, 0) = (q_1, \underline{\emptyset}, +1)$
$\delta(q_0, 1) = (q_2, \underline{\not{1}}, +1)$
$\delta(q_0, \square) = (\texttt{halt}, \square, +1)$

$\delta(q_1, x) = (q_1, x, +1)$ für $x \in \{0, 1, \underline{0}, \underline{1}\}$
$\delta(q_1, \square) = (q_3, \underline{0}, -1)$

$\delta(q_2, x) = (q_2, x, +1)$ für $x \in \{0, 1, \underline{0}, \underline{1}\}$
$\delta(q_2, \square) = (q_3, \underline{1}, -1)$

$\delta(q_3, x) = (q_3, x, -1)$ für $x \in \{0, 1, \underline{0}, \underline{1}\}$
$\delta(q_3, y) = (q_4, y, +1)$ für $y \in \{\emptyset, \not{1}, \underline{\emptyset}, \underline{\not{1}}\}$

$\delta(q_4, 0) = (q_1, \emptyset, +1)$
$\delta(q_4, 1) = (q_2, \not{1}, +1)$
$\delta(q_4, \underline{0}) = (q_5, \underline{0}, -1)$
$\delta(q_4, \underline{1}) = (q_5, \underline{1}, -1)$

$\delta(q_5, z) = (q_5, z, -1)$ für $z \in \{\emptyset, \not{1}\}$
$\delta(q_5, \underline{\emptyset}) = (q_6, 0, +1)$
$\delta(q_5, \underline{\not{1}}) = (q_6, 1, +1)$

$\delta(q_6, \emptyset) = (q_6, 0, +1)$
$\delta(q_6, \not{1}) = (q_6, 1, +1)$
$\delta(q_6, \underline{0}) = (q_6, 0, +1)$
$\delta(q_6, \underline{1}) = (q_6, 1, +1)$
$\delta(q_6, \square) = (\texttt{halt}, \square, -1)$

$\delta(q, a) = (\texttt{halt}, \square, +1)$ sonst

a) 👍 Führe M_2 auf den beiden Eingaben `0001` und `1010` aus. Gib jeweils die Konfiguration nach Ende der Berechnung, also bei Erreichen von `halt`, an.
b) 🔑 Welche Funktion berechnet M_2?

9.7 Nichtdeterminismus. Sei $L \subseteq \Sigma^*$ eine beliebige Sprache. Beweise die folgenden Aussagen:

a) 👍 Wenn es eine deterministische Turingmaschine gibt, die L entscheidet, dann gibt es auch eine nichtdeterministische Turingmaschine, die L entscheidet.
b) 🔭 Wenn es eine nichtdeterministische Turingmaschine gibt, die L entscheidet, dann gibt es auch eine deterministische Turingmaschine, die L entscheidet.

🔑 **9.8 Konstruiere eine Turingschmaschine.** Beschreibe Turingmaschinen, die die folgenden Sprachen entscheiden, bzw. die folgenden Funktionen berechnen. Die Maschine muss nicht formal angegeben werden.

a) $\{ww \mid w \in \{0, 1\}^*\}$
b) $1^n 01^m \mapsto 1^{nm}$ (falls die Eingabe nicht von der Form $1^n 01^m$ für irgendwelche n und m ist, darf die Maschine sich beliebig verhalten)
c) $1^n \mapsto$ Binärdarstellung von n

🔑 **9.9 Zwei-Stapel-Maschine.** Bei einer Zwei-Band-Maschine gibt es zwei Bänder und jedes Band hat einen eigenen Lese-/Schreibkopf. Eine *Zwei-Stapel-Maschine* ist eine Turingmaschine mit zwei Bändern und dem folgendem eingeschränkten Verhalten:

Zu jeder Zeit ist auf jedem Band jede Zelle rechts vom Kopf das Blanksymbol und jede Zelle links vom Kopf ist kein Blanksymbol. Deshalb kann sich der Kopf nur nach rechts bewegen, indem ein Symbol, das kein Blank ist, in eine Zelle mit Blank geschrieben wird. Der Kopf kann sich nur nach links bewegen, wenn die rechteste Zelle, die kein Blank ist, zu einem Blank wird. Jedes Band verhält sich also wie ein Stapel (*stack*) und die beiden Köpfe sind immer ganz oben auf dem Stapel. Um ein Überlaufen des Kopfes am linken Anfang des Bands zu vermeiden, gibt es dort ein spezielles Symbol, das nicht überschrieben werden kann. Zu Beginn enthält das erste Band den Input, wobei sich der Kopf ganz rechts über dem letzten Symbol der Eingabe befindet. Das andere Band ist, bis auf das spezielle Symbol am Anfang des Bandes, leer.

Zeige, dass jede normale Turingmaschine durch eine Zwei-Stapel-Maschine simuliert werden kann. Gegeben ist also eine normale Turingmaschine M, und du sollst eine Zwei-Stapel-Maschine M' beschreiben, die genau die gleichen Eingaben akzeptiert und verwirft wie M.