Theoretische Informatik (WS05)

Prof. Dr. Raimund Seidel
Wei Ding
Lösungsvorschlag zu Aufgabenblatt 7

Aufgabe 1

Die Aufgabenstellung versteht sich so, dass die Maschine *ein* beliebiges Auftreten von „ab“durch „ccd“ersetzt und dabei die angrenzenden Teilstrings nicht verändert.
Möchte man statt dessen *jedes* Auftreten von „ab“in dieser Weise behandeln, so genügt es, die Maschine „ein weiteres Mal über den String laufen zu lassen“(wozu ggf. ein weiterer Zustand und kleine, strukturell unbedeutende Änderungen notwendig sind).
In jedem Fall muss man an der Stelle des „ab“(mindestens) ein Zeichen einfügen und daher einen der angrenzenden Teilstrings um ein Zeichen verschieben.

Unsere TM soll auf folgende Weise arbeiten: Sie sucht das erste Autreten von „ab“und ersetzt das b durch d und das a durch c. Dann muss das Zeichen links vom a ebenfalls durch c ersetzt und in der Folge der Teilstring links vom a um ein Zeichen nach links verschoben werden (möglich wegen der Unendlichkeit des E/A-Bandes in beide Richtungen).
Es wäre auch möglich, das „ab“durch „cc“zu ersetzen und rechts davon ein b einzufügen und dabei den rechten Teilstring nach rechts zu verschieben. Wieder ein anderer Ansatz ist es, den Kopf am Anfang zu einem der Ränder des Wortes zu bewegen und das Wort in dessen Richtung zu verschieben (z.B. beginnend am linken Rand nach links), so dass beim Finden eines „ab“keine Verschiebung mehr nötig wäre. Wir beschränken uns aber auf die erste Arbeitsweise.

Im folgenden die Spezifikationen unserer einfachen TM:
$M = (\Sigma, \Gamma, \sharp, Q, s, F, \Delta\}$
$\Sigma = \{a, b, c, d\}$
$\Gamma = \{a, b, c, d, \sharp\}$
$Q = \{s, q_a, q_{ab}, q_A, q_B, q_C, q_D, q'\}, F = \{q'\}$
$\Delta = \{$
$(s, b, s, b, R), (s, c, s, c, R), (s, d, s, d, R), (s, a, q_a, a, R),$
$(q_a, a, q_a, a, R), (q_a, c, s, c, R), (q_a, d, s, d, R),$
$(q_a, b, q_{ab}, d, L), (q_{ab}, a, q_C, c, L),$

$(q_A, a, q_A, a, L), (q_A, b, q_B, a, L), (q_A, c, q_C, a, L), (q_A, d, q_D, a, L),$
$(q_B, a, q_A, b, L), (q_B, b, q_B, b, L), (q_B, c, q_C, b, L), (q_B, d, q_D, b, L),$
$(q_C, a, q_A, c, L), (q_C, b, q_B, c, L), (q_C, c, q_C, c, L), (q_C, d, q_D, c, L),$
$(q_D, a, q_A, d, L), (q_D, b, q_B, d, L), (q_D, c, q_C, d, L), (q_D, d, q_D, d, L),$

$(q_A, \sharp, q', a, B), (q_B, \sharp, q', b, B), (q_C, \sharp, q', c, B), (q_D, \sharp, q', d, B)$
$\}$

Wirkungsweise:
Die Regeln der ersten Zeile unter Δ bewegen den Kopf nach rechts, bis ein a gefunden wird, und lassen die Maschine dann in den Zustand q_a wechseln. Durch die Regeln in der zweiten Zeile wechselt die TM zurück in den Startzustand, falls rechts vom a ein c oder d steht, und verbleibt in q_a, falls ein a folgt.

Sollte aber ein b folgen, so wechselt die TM in den Zustand q_{ab} (erste Regel in der dritten Zeile), ersetzt dieses b dabei durch ein d und bewegt den Kopf nach links. Nun wird das dort stehende a durch ein c ersetzt, in den Zustand q_C gewechselt und der Kopf wieder nach links bewegt. Der Zustand q_C sorgt dafür, dass links ein weiteres c geschrieben wird, so dass wir dann „ccd"geschrieben haben (s. nächser Abschnitt).
Die folgenden 16 Regeln und die vier Zustände q_A, q_B, q_C und q_D arbeiten alle gleich: Sie verschieben das Wort Zeichenweise nach links, indem sie das zuvor gelesene Zeichen (bei q_A z.B. a) schreiben, dabei in den zum aktuellen Zeichen passenden Zustand wechseln (z.b. bei einem c in q_C) und den Kopf wieder nach links bewegen.
Dies passiert so lange, bis ein ♯ auftritt (letzte Zeile). Dann wird nur noch das zuletzt gelesene Zeichen geschrieben und in den Endzustand q' gewechselt. Der Kopf steht nun über der linkesten beschriebenen Bandzelle und ist im gewünschten Zustand.

Falls es im Wort kein „ab"gibt, sucht die Maschine vergebens, d.h. der Kopf wandert mit den ersten vier Regeln bis zum rechten Rand und landet auf einem ♯. Da unsere Maschine für diesen Fall keine Regeln hat, wird sie halten und sich im Zustand s oder q_a befinden, aber niemals q' erreichen.

Aufgabe 2

a)

Formale Spezifikation eines 2NKA:

$M = (Q, s, F, \Sigma, \Gamma, €, \Delta)$

Q	Zustandsmenge
$s \in Q$	Startzustand
$F \subseteq Q$	Menge der Endzustände
Σ	Eingabealphabet
Γ	Arbeitsalphabet
$€ \in \Gamma$	Kellerbodensymbol
Δ	Übergangsregeln

Dabei gilt für die Menge der **Übergangsregeln**:

$\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma^* \times \Gamma^* \times Q \times \Gamma^* \times \Gamma^*$

Die Regeln sind also von der Form (q, a, U, V, r, W, X) mit

q: aktueller Zustand
a: aktuelles Zeichen
U: aktueller Inhalt Keller 1
V: aktueller Inhalt Keller 2
r: neuer Zustand
W: neuer Inhalt Keller 1
X: neuer Inhalt Keller 2

Eine Regel $(q, a, AU, BV, r, CU, DV) \in \Delta$ bedeutet also: Befindet sich der Automat in Zustand q, das Eingabeband liest ein a und die obersten Symbole von Keller 1 und Keller 2 sind ein A bzw ein B, so geht er nun in Zustand r über und ersetzt die obersten Kellersymbole durch C bzw D.

Die aktuelle **Konfiguration** eines 2NKA ist charakterisiert durch den aktuellen Zustand, in dem er sich befindet, das noch zu lesende (Teil-)Wort sowie die aktuellen Kellerinhalte von Keller 1 und Keller 2.

Formal also: $k = (q, w, X, Y)$ mit $q \in Q, w \in \Sigma^*, X, Y \in \Gamma^*$

Demzufolge ist also der **Konfigurationsraum** : $K_M = Q \times \Sigma^* \times \Gamma^* \times \Gamma^*$

Der Automat soll nun genau dann von der Konfiguration $k_1 = (q_1, aw, AX, BY)$ zur Konfiguration $k_2 = (q_2, w, CX, DY)$ kommen, wenn eine Regel $(q_1, a, AX, BY, q_2, CX, DY)$ vorhanden ist.

Wir können also die **Rechenschrittrelation** $k_1 \longmapsto_M k_2$ definieren:

$(q_1, aw, AX, BY) \longmapsto_M (q_1, a, CX, DY, q_2) :\Longleftrightarrow (q_1, a, AX, BY, q_2, CX, DY) \in \Delta$

Bevor der Automat mit dem Lesen eines Wortes w beginnt, hat er die **Startkonfiguration** $k_0 = (s, w, €, €)$. Wenn wir uns nach dem Lesen des Wortes in einem Endzustand befinden, so soll das Wort akzeptiert werden. **Endkonfigurationen** sind also alle Konfiguration $k_n = (f, \epsilon, U, V)$ mit $f \in F$ (sowie $U, V \in \Gamma^*$).

Ein Wort $w \in \Sigma^*$ soll also genau dann **akzeptiert** werden, wenn eine Folge von $n+1$ Konfigurationen $k_0, \ldots, k_n$ existiert, mit $k_0 = (s, w, €, €)$ und $k_n = (f, \epsilon, U, V)$ mit $f \in F$, so dass

$k_0 \longmapsto_M k_1 \longmapsto_M \ldots \longmapsto_M k_{n-1} \longmapsto_M k_n$ (also: $k_0 \longmapsto_M^* k_n$)

2NKA-Sprachen sind nun alle Sprachen, für die ein 2NKA existiert, der genau diese Sprache akzeptiert.

b)

Wir betrachten wieder das Beispiel $L_{abc} = \{a^n b^n c^n | n \in \mathbb{N}\}$

In der Vorlesung wurde bereits gezeigt, dass diese Sprache keine NKA-Sprache ist. Wir können jedoch einen 2NKA angeben, der diese Sprache akzeptiert:

Dieser liest zunächst alle a's und schreibt jeweils ein Zeichen in den 1.Keller. Anschliessend liest er die b's, wobei er gleichzeitig ein Zeichen aus dem 1.Keller nimmt und auf den 2. Keller schreibt. Ist nun der 1.Keller leer, so liest er ab jetzt nur noch c's und nimmt jeweils ein Zeichen aus dem 2.Keller. Ist nach dem Lesen des Wortes der 2.Keller leer, so geht er in den Endzustand.

Formal: $M = (Q, s, F, \Sigma, \Gamma, €, \Delta)$

$Q = \{s, a, b, c, f\}$
$F = \{f\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{X, €\}$

$\Delta = \{$ (s,ϵ, €, €, f, €, €)
(s,a,U, €, s, XU, €),
(s,b, XU, €,b, U, X),
(b,b, XU, V,b, U, XV),
(b,c, €, c, XV, €, V),
(c,c, €, c, XV, €, V),
(c,ϵ, €, €, f, €, €)$\}$

c)

Behauptung: Die 2NKA-Sprachen sind genau die Sprachen die von Turing-Maschinen akzeptiert werden.

Beweis:

1.) L TM-Sprache $\Rightarrow$ L 2NKA-Sprache:

Sei L eine Sprache die von einer TM M akzeptiert wird. Sei oBdA M eine Turingmaschine mit nur einem Band. Wir wollen einen 2NKA M' konstruieren, der M' simuliert.

Idee: Simuliere das Arbeitsband mit den beiden Kellern: Speichere die bis jetzt gelesenen Zeichen in Keller 1, die noch zu lesenen Zeichen in Keller 2. Das oberste Symbol von Keller 2 ist also das Symbol, auf das der Lese-/Schreibkopf der TM M zeigen würde. Durch eine geeignete Simulation der Übergangsregeln der TM müssen wir dafür sorgen, dass diese Invariante immer erfüllt ist.

Der 2NKA M' verfügt über einen Startzustand s'. In diesem lesen wir zunächst alle Zeichen durch und schreiben diese in Keller 2, anschliessend übertragen wir diese in Keller 1. Erst nachdem wir das erledigt haben, gehen wir in den Zustand s (den alten Startzustand der TM) und beginnen mit der eigentlichen Simulation.

Um diese Startsituation herzustellen, benötigen wir also folgende Regeln:

$(s', x, U, V, xU, V, s') \in \Delta'$.
$(s', \epsilon, xU, V, U, xV, s') \in \Delta'$
$(s', \epsilon, €, \mathrm{V}, €, \mathrm{V}, s) \in \Delta'$

Die Regeln der Turing-Maschine werden nun folgendermassen simuliert:

Betrachten wir eine Regel $(q_1, x, q_2, y, b) \in \Delta$. Die Turingmaschine wechselt also, falls sie sich im Zustand q_1 befindet und ein x liest, in den Zustand q_2 und ersetzt das x durch ein y. Der Eingabekopf bleibt auf der Stelle stehen. Um diese Regel mit einem 2NKA zu simulieren, müssen wir also, falls das oberste Zeichen von Keller 2 ein x ist, dieses durch ein y ersetzen. Also: $(q_1, \epsilon, U, xV, q_2, U, yV) \in \Delta')$

Soll sich der Eingabekopf nach links bewegen, so müssen wir zusätzlich das oberste Element des Kellers 1 auf den Keller 2 legen. Analog legen wir bei einer Rechtsbewegung das oberste Element des Kellers 2 auf den Keller 1. Also:

$(q_1, x, q_2, y, l) \in \Delta) \Longleftrightarrow (q_1, \epsilon, aU, xV, q_2, U, ayV) \in \Delta'$

$(q_1, x, q_2, y, r) \in \Delta) \Longleftrightarrow (q_1, \epsilon, U, xV, q_2, yU, V) \in \Delta'$

Damit ein Wort von der ursprünglichen TM akzeptiert wird, müssen zwei Bedingungen erfüllt sein: 1. Wir müssen in einem Endzustand sein. 2. Das Wort muss zuende gelesen worden sein. Die zweite Regel entspricht in unserem 2NKA einem leeren 2.Keller. Also führen wir einen

Super-Endzustand f' ein, in den wir wechseln, falls der 2.Keller leer ist und wir uns in einem „alten" Endzustand befinden.

also: $\forall f \in F : (f, \epsilon, X, €, f', X, e) \in \Delta'$.

Insgesamt gilt also für den konstruierten 2NKA M':

$M' = (Q', s', F', \Sigma', \Gamma', €, \Delta')$

$Q' = Q \cup \{s', f'\}$
s'
$F' = \{f'\}$
$\Sigma' = \Gamma$
$\Gamma' = \Gamma \cup \{€\}$

Δ': $(s', x, U, V, xU, V, s') \in \Delta'$
$(s', \epsilon, xU, V, U, xV, s') \in \Delta'$
$(s', \epsilon, €, \mathrm{V}, €, \mathrm{V}, s) \in \Delta'$
$\forall (q_1, x, q_2, y, b) \in \Delta : (q_1, \epsilon, U, xV, q_2, U, yV) \in \Delta'$
$\forall (q_1, x, q_2, y, l) \in \Delta : (q_1, \epsilon, aU, xV, q_2, U, ayV) \in \Delta'$
$\forall (q_1, x, q_2, y, r) \in \Delta : (q_1, \epsilon, U, xV, q_2, yU, V) \in \Delta'$
$\forall f \in F : (f, \epsilon, X, €, f', X, €) \in \Delta'$

2.) L 2NKA-Sprache $\Rightarrow$ L TM-Sprache:

Sei L eine Sprache die von einem 2NKA M akzeptiert wird. Wir wollen eine TM M' konstruieren, die M simuliert.

In der Vorlesung wurde gezeigt, dass Turing-Maschinen mit einem Band gleichmächtig zu Turing Maschinen mit beliebig vielen Bändern sind. Sei also M' eine TM mit einem Eingabeband und 2 Arbeitsbändern.

Wir können nun die beiden Keller leicht mit den beiden Arbeitsbändern simulieren. Wir schreiben nun jeweils ein € auf das 1. und 2. Arbeitsband. Die Zeichen links von diesem €-Zeichen identifizieren wir mit dem Inhalt des entsprechenden Kellers. Das Ersetzen eines Zeichens auf dem n-ten Keller entspricht also dem Ersetzen des jeweiligen Zeichens auf dem n-ten Arbeitsband. Wollen wir auf dem Keller hinzufügen oder löschen, so müssen wir im entsprechenden Arbeitsband zunächst nach links bzw rechts gehen und können dann mit einer ϵ-Regel das entsprechende Zeichen schreiben.

Möchten wir mit der Turingmaschine eine ϵ-Regel des 2NKA simulieren, so lassen wir in der entsprechenden Regel der Turingmaschine den Eingabekopf an der aktuellen Position verweilen, ansonsten bewegen wir ihn nach rechts.

Auf diese Weise kann also eine TM gefunden werden, die genau die Sprache L akzeptiert.

Aus 1.) und 2.) folgt: L 2NKA-Sprache $\Leftrightarrow$ L TM-Sprache