Übungen zur Vorlesung Grundbegriffe der Theoretischen Informatik

technische universität dortmund

Thomas Schwentick

Gaetano Geck, Lutz Oettershagen, Christopher Spinrath, Marco Wilhelm

Lehrstuhl 1 Logik in der Informatik

SoSe 2018 | **Übungsblatt 8** | **05.06.2018**

Aufgabe 8.1 [WHILE- und GOTO-Programme] — 6 Punkte

a) Sei P das folgende WHILE-Programm und $f_P : \mathbb{N}_0 \rightharpoonup \mathbb{N}_0$ die von P berechnete partielle Funktion.

```
1   x2 := 1;
2   x4 := 2;
3   WHILE x1 ≠ 0 DO
4     x3 := x2;
5     WHILE x3 ≠ 0 DO
6       x3 := x3 ∸ 1;
7       x2 := x2 + 1
8     END;
9     x1 := x1 ∸ 1;
10    x4 := x4 ∸ 1
11  END;
12  WHILE x4 ≠ 0 DO
13    x4 := x4 + 1
14  END;
15  x1 := x2
```

(3 Punkte)

(i) Bestimmen Sie die Funktionswerte $f_P(1)$ und $f_P(2)$, falls diese definiert sind, oder begründen Sie andernfalls, warum sie nicht definiert sind. **[1 Punkt]**

(ii) Geben Sie die Funktion f_P formal an. Bestimmen Sie außerdem den Definitionsbereich $D(f_P)$ sowie den Wertebereich $W(f_P)$ von f_P. **[2 Punkte]**

b) Begründen Sie, dass die Funktion $\max : \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0$ mit der üblichen Semantik GOTO-berechenbar ist. Geben Sie dazu ein GOTO-Programm an und begründen Sie stichhaltig, dass Ihr Programm die Funktion max berechnet. **(3 Punkte)**

Lösung:

a) **(i)** Die Funktion f_P ist an der Stelle 1 *nicht definiert*, d.h. es gilt $f_P(1) = \perp$. Vor dem ersten Durchlauf der ersten (äußeren) WHILE-Schleife ist der Speicherinhalt $1, 1, 0, 2, \ldots$ (x_1 hat also den Wert 1 und x_4 den Wert 2), da die Eingabe 1 der Variablen x_1 zugewiesen wird und die ersten beiden Zeilen den Variablen x_2 und x_4 die Werte 1 und 2 zuweisen. In in der Schleife wird x_1 in Zeile 9 um eins dekrementiert; ansonsten kommt x_1 nicht im Schleifenrumpf vor. Nach dem ersten Durchlauf ist der Variablen x_1 also der Wert 0 zugewiesen und das Programm wird in Zeile 12 fortgesetzt. Außerdem wird x_4 in der ersten Schleife ebenfalls um eins dekrementiert. Das Programm „befindet“ sich also in Zeile 12 und x_4 ist der Wert $2-1=1$ zugewiesen. Der Schleifenrumpf der WHILE-Schleife wird also ausgeführt.

Es ist nun einfach zu sehen, dass die Variable unter diesen Bedingungen niemals den Wert 0 annimmt, denn sie wird im Schleifenrumpf stets inkrementiert. Das Programm hält also nie an, und damit gilt, wie behauptet, $f_P(1) = \bot$.

An der Stelle 2 ist die Funktion f_P hingegen *definiert*. Es gilt $f_P(2) = 4$. Analog zur obigen Argumentation wird die erste äußere Schleife zweimal durchlaufen, denn in jedem Durchlauf wird x_1 um eins dekrementiert. Ebenso wird x_4 um zwei dekrementiert. Die innere Schleife in den Zeilen 5 bis 8 addiert gerade die Speicherinhalte von x_2 und x_3 (vergleiche Vorlesung), wobei x_3 zuvor der Wert von x_2 zugewiesen wird. Der Effekt des Schleifenrumpfes bzgl. x_2 ist also gerade $x_2 := x_2 + x_2$, der Wert der Variablen x_2 wird in jedem Durchlauf also verdoppelt. Nach zwei Durchläufen hat die Variable x_2 also den Wert 4, weil der Inhalt in Zeile 1 mit 1 initialisiert wird. Der Schleifenrumpf der Schleife in den Zeilen 12 bis 14 wird niemals betreten, weil der Wert von x_4 gerade $2 - 2 = 0$ ist. In der letzten Zeile wird x_1 schließlich der Wert von x_2 zugewiesen; und dieser Wert, also 4, ist damit die Ausgabe.

(ii) Sei $n \in \mathbb{N}_0$ beliebig. Wie wir bereits in (i) argumentiert haben, wird bei Eingabe n die erste äußere Schleife genau n mal durchlaufen, wobei der Wert von x_4 jeweils um eins dekrementiert wird (falls der Wert nicht bereits 0 ist) und der Wert von x_2 jeweils verdoppelt wird. Der Effekt der ersten Schleife ist daher $x_1 = 0$, $x_2 = 2^n$ und $x_4 = \max(0, 2 - n)$ (bemerke, dass der Wert von x_4 initial auf 2 gesetzt wird). Im Fall $n < 2$ nimmt x_4 in Zeile 12 also einen Wert ungleich 0 an, und das Programm terminiert nicht. Andernfalls wird der Wert von x_2 „ausgegeben“, also gerade 2^n. Insgesamt ergibt sich die folgende Funktionsvorschrift.

$$f_P : n \mapsto \begin{cases} 2^n, & \text{falls } n \geq 2 \\ \bot, & \text{sonst} \end{cases}$$

Der Definitionsbereich ist damit $D(f_P) = \{n \in \mathbb{N}_0 \mid n \geq 2\}$ und der Wertebereich $W(f_P) = \{2^n \mid n \in \mathbb{N}_0, n \geq 2\}$.

b) Sei P' das folgende GOTO-Programm.

```
1: x3 := x1;
2: x4 := x2;
3: IF x3 = 0 THEN GOTO 7;
4: x3 := x3 ∸ 1;
5: x4 := x4 ∸ 1;
6: IF x5 = 0 THEN GOTO 3;
7: IF x4 = 0 THEN GOTO 9;
8: x1 := x2;
9: HALT
```

Wir behaupten, dass P' die Funktion max berechnet, also, dass $f_{P'} = \max$ gilt, wobei wir $f_{P'}$ als zweistellige Funktion auffassen.

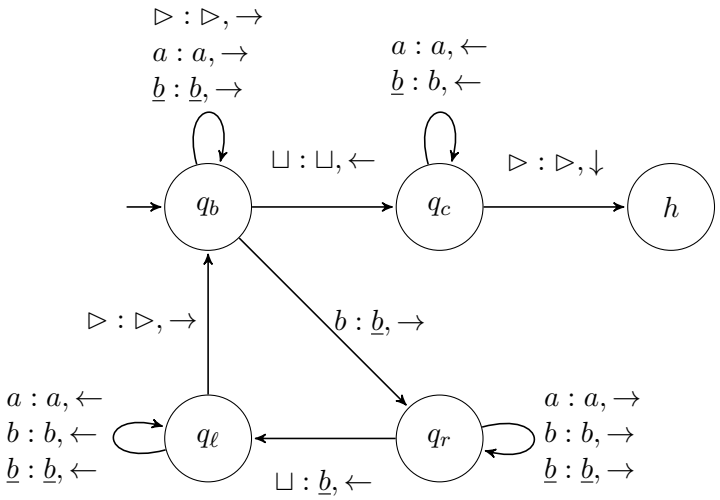
Die beiden Eingabewerte werden initial den Variablen x_1 und x_2 zugewiesen. Da wir einen der beiden Werte später „ausgeben“ müssen, nutzen wir die Variablen x_3 und x_4 zum rechnen, und lassen x_1 und x_2 (zunächst) unberührt. Die Bedingung $x_5 = 0$ in der bedingten Sprunganweisung in Zeile 6 ist immer wahr, weil x_5 sonst nicht im Programm vorkommt, und der Inhalt von x_5 daher immer 0 ist. Erreicht eine Berechnung also Zeile 6, so wird die Berechnung stets in Zeile 3 fortgesetzt. Damit werden die Inhalte von x_3

und x_4 solange dekrementiert, bis der Inhalt von x_3 gleich 0 ist, und damit die Bedingung der Sprunganweisung in Zeile 3 wahr ist (und Zeile 6 übersprungen wird). Der Effekt der ersten sechs Zeilen ist daher $x_4 = \max(0, x_2 - x_1)$, denn es wird $x_4 = x_4 \dot{-} x_3$ berechnet und x_3 und x_4 werden initial die Werte von x_1 und x_2 zugewiesen.

Die Berechnung befindet sich nun in Zeile 7. Ist die Bedingung der Sprunganweisung wahr, so ist der Inhalt von x_4 gleich 0. Dies kann aber nur sein, wenn der Inhalt von x_2 initial kleiner oder gleich dem Inhalt von x_1 war. Folgerichtig geht die Berechnung in Zeile 9 weiter und endet; die Ausgabe ist der Inhalt von x_1, welcher größer ist, als der Inhalt von x_2. Andernfalls wird in Zeile 8 der Inhalt von x_1, welcher in diesem Falle kleiner dem Inhalt von x_2 ist, auf den Inhalt von x_2 gesetzt und dann in Zeile 9 angehalten – auch in diesem Fall wird also die größere Eingabezahl ausgegeben.

Aufgabe 8.2 [Turingmaschinen: Interpretation] **4 Punkte**

Gegeben sei die folgende Turingmaschine M mit dem Arbeitsalphabet $\{a, b, \underline{b}, \sqcup, \triangleright\}$.



a) Geben Sie die ersten zehn Konfigurationen sowie, falls existent, die erste Konfiguration mit aktuellem Zustand q_c und die Haltekonfiguration der Berechnung von M auf der Eingabe bab an. **(1 Punkt)**

b) Beschreiben Sie die Arbeitsweise der Turingmaschine M für beliebige Eingaben über dem Eingabealphabet $\{a, b\}$. Gehen Sie dabei insbesondere auf die Bedeutung der einzelnen Zustände ein. Geben Sie außerdem die von M berechnete Funktion $f_M : \{a, b\}^* \rightharpoonup \{a, b\}^*$ formal an. **(3 Punkte)**

Lösung:

a) Um die Berechnung vollständig nachvollziehen zu können, geben wir hier alle Konfigurationen der Berechnung an. Gefordert sind laut Aufgabenstellung nur die farbig hervorgehobenen Konfigurationen. Wir notieren die Konfigurationen mit dreistelligen String-Zeigerbeschreibungen.

$$
\begin{aligned}
(q_b, (\varepsilon, \triangleright, bab)) &\vdash (q_b, (\triangleright, b, ab)) \\
&\vdash (q_r, (\triangleright\underline{b}, a, b)) \\
&\vdash (q_r, (\triangleright\underline{b}a, b, \varepsilon)) \\
&\vdash (q_r, (\triangleright\underline{b}ab, \sqcup, \varepsilon)) \\
&\vdash (q_\ell, (\triangleright\underline{b}a, b, \underline{b})) \\
&\vdash (q_\ell, (\triangleright\underline{b}, a, b\underline{b})) \\
&\vdash (q_\ell, (\triangleright, \underline{b}, ab\underline{b})) \\
&\vdash (q_\ell, (\varepsilon, \triangleright, \underline{b}ab\underline{b})) \\
&\vdash (q_b, (\triangleright, \underline{b}, ab\underline{b})) \\
&\vdash (q_b, (\triangleright\underline{b}, a, b\underline{b})) \\
&\vdash (q_b, (\triangleright\underline{b}a, b, \underline{b})) \\
&\vdash (q_r, (\triangleright\underline{b}a\underline{b}, \underline{b}, \varepsilon)) \\
&\vdash (q_r, (\triangleright\underline{b}ab\underline{b}, \sqcup, \varepsilon)) \\
&\vdash (q_\ell, (\triangleright\underline{b}a\underline{b}, \underline{b}, \underline{b})) \\
&\vdash (q_\ell, (\triangleright\underline{b}a, \underline{b}, \underline{b}\underline{b})) \\
&\vdash (q_\ell, (\triangleright\underline{b}, a, \underline{b}\underline{b}\underline{b})) \\
&\vdash (q_\ell, (\triangleright, \underline{b}, a\underline{b}\underline{b}\underline{b})) \\
&\vdash (q_\ell, (\varepsilon, \triangleright, \underline{b}a\underline{b}\underline{b}\underline{b})) \\
&\vdash (q_b, (\triangleright, \underline{b}, a\underline{b}\underline{b}\underline{b})) \\
&\vdash (q_b, (\triangleright\underline{b}, a, \underline{b}\underline{b}\underline{b})) \\
&\vdash (q_b, (\triangleright\underline{b}a, \underline{b}, \underline{b}\underline{b})) \\
&\vdash (q_b, (\triangleright\underline{b}a\underline{b}, \underline{b}, \underline{b})) \\
&\vdash (q_b, (\triangleright\underline{b}a\underline{b}\underline{b}, \underline{b}, \varepsilon)) \\
&\vdash (q_b, (\triangleright\underline{b}a\underline{b}\underline{b}\underline{b}, \sqcup, \varepsilon)) \\
&\vdash (q_c, (\triangleright\underline{b}a\underline{b}\underline{b}, \underline{b}, \sqcup)) \\
&\vdash (q_c, (\triangleright\underline{b}a\underline{b}, \underline{b}, b\sqcup)) \\
&\vdash (q_c, (\triangleright\underline{b}a, \underline{b}, bb\sqcup)) \\
&\vdash (q_c, (\triangleright\underline{b}, a, bbb\sqcup)) \\
&\vdash (q_c, (\triangleright, \underline{b}, abbb\sqcup)) \\
&\vdash (q_c, (\varepsilon, \triangleright, babbb\sqcup)) \\
&\vdash (h, (\varepsilon, \triangleright, babbb\sqcup))
\end{aligned}
$$

b) Die von M berechnete Funktion lautet

$$f_M : \{a, b\}^* \mapsto \{a, b\}^*, w \mapsto wb^i \text{ mit } i = \#_b(w).$$

Die Turingmaschine hängt also genauso viele b-Symbole an das Eingabewort an, wie b-Symbole in der Eingabe vorkommen.

Wir betrachten zunächst das „Dreieck“ der Turingmaschine M mit den Zuständen q_b, q_r und q_ℓ. Beginnend im Startzustand q_b liest M solange Zeichen und bewegt ihren Zeiger nach rechts, bis ein b (oder $\sqcup$) gelesen wird. Der Zustand q_b wird also dafür genutzt,

dass nächste b rechts von der aktuellen Zeigerposition zu finden. Das gefundene b wird beim Übergang in den Zustand q_r mit einem Strich markiert (überschrieben mit $\underline{b}$). Anschließend läuft die Machine im Zustand q_r nach rechts bis zum ersten $\sqcup$ und hängt beim Übergang zum Zustand q_ℓ ein $\underline{b}$ an den bisherigen String an. Im Zustand q_ℓ läuft die Maschine danach wieder ganz nach links in die ursprüngliche Ausgangsposition zurück (in den Zustand q_b und der Zeiger zeigt auf das erste Symbol). Die Zustände q_r und q_ℓ sind also dafür zuständig ganz nach rechts bzw. links zu laufen, sowie dafür ein $\underline{b}$-Symbol anzuhängen.

Dieses Prozedere wiederholt sich, bis keine b-Symbole mehr übrig sind – in diesem Fall hat M alle b-Symbole durch $\underline{b}$-Symbole ersetzt und hat *genauso viele* $\underline{b}$-Symbole an die (modifizierte) Eingabe angehängt. Außerdem wird M nun im Zustand q_b nach rechts laufen bis es auf das erste $\sqcup$-Symbol trifft und damit in den Zustand q_c übergehen. Der Zustand q_c ist ein Hilfszustand, der „aufräumt“ bevor die Maschine anhält: die Maschine läuft nach links und ersetzt dabei alle $\underline{b}$-Symbole durch b-Symbole. Es werden also alle angehängten $\underline{b}$-Symbole durch b-Symbole ersetzt und die „ursprüngliche“ Eingabe wird wiederhergestellt (ohne die angefügten b-Symbole zu entfernen). Wenn M dann auf dem $\triangleright$-Symbol steht, ist der aktuelle String also tatsächlich $\triangleright w b^i$ mit $i = \#_b(w)$ (bei Eingabe $w \in \{a,b\}^*$). Insbesondere ist die Maschine im Haltezustand und steht ganz links, damit ist der Funktionswert an der Stelle w also gerade wb^i mit $i = \#_b(w)$, wie behauptet.

Aufgabe 8.3 [Turingmaschinen: Modellierung] 5 Punkte

a) Die Distanz zweier Wörter $u, v \in \{a,b\}^*$ mit $|u| = |v|$ sei wie folgt definiert:

$$\mathrm{dist}(u,v) = |\{i \in \{1,\ldots,|u|\} \mid u[i] \neq v[i]\}|.$$

Die Distanz von u und v ist also die Anzahl der Stellen an denen sich u und v unterscheiden.

Entwerfen Sie eine Turingmaschine, welche die Funktion

$$f : \{a,b,\$\}^* \rightharpoonup \{a,b,\$\}^* \text{ mit } f(u\$v\$) = u\$v\$a^{\mathrm{dist}(u,v)}$$

für alle $u, v \in \{a,b\}^*$ mit $|u| = |v|$ berechnet. Die Turingmaschine soll also für ein Eingabewort der Form $u\$v\$$ mit $|u| = |v|$ die Anzahl der Stellen zählen an denen sich u und v unterscheiden und entsprechend oft das Zeichen a an das Eingabewort anfügen. Für Wörter über dem Eingabealphabet $\{a,b,\$\}$, die nicht die beschriebene Form haben, soll die Funktion f nicht definiert sein.

Geben Sie die Turingmaschine formal, *ohne* die Transitionsfunktion zu spezifizieren, *und* als Diagramm an. Beschreiben Sie die Arbeitsweise Ihrer Turingmaschine und erläutern Sie die Bedeutung der einzelnen Zustände. **(4 Punkte)**

Hinweis

Nutzen Sie zusätzliche Symbole, um die Zeichen des Eingabewortes zu markieren, die bereits abgehandelt wurden. Ob die Eingabe die beschriebene Form hat, kann „on-the-fly“ überprüft werden.

b) Beschreiben Sie, wie Ihre Turingmaschine aus Aufgabenteil a) abgewandelt werden müsste, sodass die enstehende Turingmaschine genau die Wörter, die in der Sprache

$$L = \{w\$w\$ \mid w \in \{a,b\}^*\}$$

sind, *akzeptiert*. **(1 Punkt)**

Lösung:

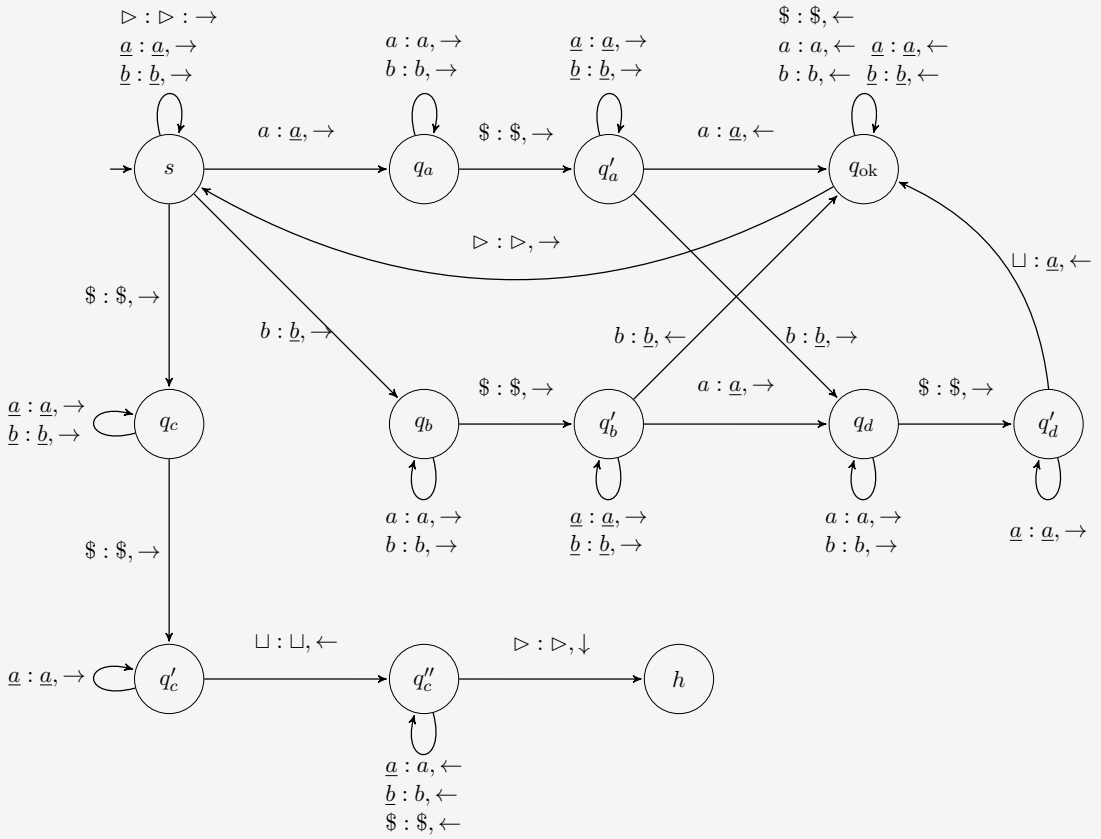
a) Unsere Konstruktionsidee für die geforderte Turingmaschine ist wie folgt: Um bei Eingabe $u\$v\$$ die Distanz von u und v zu berechnen genügt es für jede Position $1 \leq i \leq |u|$ die Symbole $u[i]$ und $v[i]$ zu vergleichen; sind die Symbole unterschiedlich, so kann (unabhängig von der restlichen Eingabe) ein a hinter das zweite Trennsymbol \$ angehängt werden (die Distanz wird unär gezählt); aus technischen Gründen wird die Maschine tatsächlich zunächst nur markierte a-Symbole anhängen und diese beim Aufräumen ersetzen (dazu später mehr). Dazu geht die Turingmaschine wie folgt vor:

- Angenommen alle Positionen $j < i$ wurden bereits verglichen (gegebenenfalls wurden a-Symbole angehängt) und sowohl in u als auch v sind alle Positionen $j < i$ markiert.
- Um die Symbole an Position i zu vergleichen läuft die Maschine zu dem Zeichen links vor dem ersten Trennsymbol \$, welches noch nicht markiert ist und das Symbol mit dieser Eigenschaft ist, welches am weitesten links steht. Dieses Symbol muss $u[i]$ sein.
- Die Maschine merkt sich nun das Symbol und markiert es.
- Anschließend läuft die Turingmaschine hinter das erste Trennsymbole und sucht dort das am weitesten links stehende Zeichen, welches noch nicht markiert ist. Dieses Zeichen ist $v[i]$ und die Turingmaschine kann nun $u[i]$ und $v[i]$ vergleichen.
- Nachdem auch $v[i]$ markiert wurde, hängt die Maschine gegebenfalls ein a an (sie läuft ganz nach rechts auf das erste $\sqcup$-Zeichen und schreibt ein a), und läuft dann ganz nach links.
- Nun ist die Turingmaschine in der gewünschten Ausgangsposition, um die Symbole an Position $i + 1$ zu vergleichen.

Konkret sei M die Turingmaschine $M = (Q, \Gamma, \delta, s)$ mit

- Zustandsmenge $Q = \{s, q_a, q'_a, q_b, q'_b, q_{\text{ok}}, q_d, q'_d, q_c, q'_c, q''_c\}$,
- Arbeitsalphabet $\Gamma = \{a, b, \$\} \cup \{\underline{a}, \underline{b}\} \cup \{\triangleright, \sqcup\} = \{a, b, \$, \underline{a}, \underline{b}, \triangleright, \sqcup\}$,
- Transitionsfunktion δ und Startzustand s.

Die Transitionsfunktion δ ist durch das nachfolgende Diagramm gegeben, wobei wir die Konvention nutzen, dass nicht eingezeichnete Transitionen in den Zustand nein führen.



Die Zustände der Turingmaschine haben die folgenden Bedeutungen (vergleiche obige Konstruktionsidee):

- Der Startzustand s ist dafür zuständig das erste nicht markierte Symbol vor dem ersten Trennzeichen \$ zu finden. Wird keines gefunden (es wird das Trennzeichen \$ gelesen), so geht die Turingmaschine in die „Aufräumphase“ über, die mit den Zuständen q_c, q_c' und q_c'' realisiert ist. Ansonsten wird das gefundene Symbol (also ein a oder b) im Zustandsraum gespeichert indem in den Zustand q_a bzw. q_b übergegangen wird. Außerdem wird das Symbol markiert (mit dem Zeichen $\underline{a}$ bzw. $\underline{b}$ überschrieben).
- In den Zuständen q_a und q_a' wissen wir, dass in dem Teilwort vor dem Trennsymbol zuletzt ein a gelesen (und markiert) wurde. Die Turingmaschine läuft zunächst zum Trennsymbol (Zustand q_a) und dann zum ersten nicht-markierten Symbol rechts vom Trennsymbol (Zustand q_a'). Nun steht der Zeiger auf dem Symbol, mit dem das gespeicherte Symbol verglichen werden muss. Handelt es sich um ein a kann das Symbol einfach markiert werden und der nächste Vergleich gestartet werden, denn die Zeichen sind gleich.
- Dazu geht M in den Zustand q_{ok} über, der dafür zuständig ist, ganz nach links zu laufen und in den Startzustand überzugehen.
- Sind die Zeichen unterschiedlich, muss ein a hinter das zweite Trennsymbol angehängt werden. Dafür sind die Zustände q_d und q_d' zuständig. Da der Zeiger sicher an einer Position hinter dem ersten Trennsymbol steht, genügt es hinter das zweite Trennsymbol zu laufen (Zustand q_d) und danach das erste ⊔-Symbol mit einem $\underline{a}$ zu

überschreiben (Zustand q'_d; alle $\underline{a}$-Symbole werden später durch a-Symbole ersetzt). Danach kann wie im ersten Fall in den Zustand q_{ok} übergegangen werden, um den nächsten Vergleich zu beginnen.

- Die Zustände q_b und q'_b verhalten sich analog zu q_a und q'_a, wobei die Rollen von a und b vertauscht sind.
- Die Zustände q_c, q'_c und q''_c sind schließlich für das „Aufräumen“ zuständig. Es müssen alle $\underline{a}$- und $\underline{b}$-Symbole durch a- bzw. b-Symbole ersetzt werden (die Markierungen müssen entfernt werden) und am Ende muss, damit der Funktionswert für das Eingabewort definiert ist, ganz nach links gelaufen werden und in den Zustand h übergegangen werden. In den Zuständen q_c und q'_c läuft die Maschine dazu zunächst nach rechts, um die Symbole zu ersetzen, und prüft dabei auch, ob genau zwei Trennsymbole in dem Eingabewort vorkammen. Im Zustand q''_c wird schließlich noch der Zeiger auf den linken Rand gesetzt.

Um zu verifizieren, dass die Eingabe die Form $u\$v\$$ mit $|u| = |v|$ hat, nutzen wir die Konvention, das alle nicht eingezeichneten Transitionen in den Zustand nein führen. In diesem Fall ist die Funktion f_M für das jeweilige Eingabewort dann, wie gewünscht, nicht definiert.

- In den Zuständen q'_a und q'_b wird sichergestellt, dass $|u| \leq |v|$ gilt. In diesen Zuständen wissen wir, dass der Zeiger hinter dem ersten Trennzeichen \$ steht (aber vor dem zweiten, falls existent) und ein noch nicht markiertes Zeichen vor dem zweiten Trennsymbol gefunden werden muss, mit dem das aktuell gemerkte Symbol aus dem Teilwort u verglichen werden muss. Dieses Zeichen wird mit den einzigen eingezeichneten, in einen anderen Zustand führenden, Transitionen „gefunden“. Wenn es ein solches Zeichen nicht gibt (also ein Trennzeichen \$ oder ein $\sqcup$ gelesen wird), so muss $|u| > |v|$ gelten, und durch die obige Konvention ist sichergestellt, dass die Funktion der Maschine für die gegebene Eingabe nicht definiert ist, denn es sind keine entsprechenden Transitionen eingezeichnet.
- Ähnlich wird im Zustand q_c sichergestellt, dass $|u| \geq |v|$ gilt. Würde $|u| < |v|$ gelten, so gebe es beim Aufräumen in dem aktuellen String ein nicht markiertes Zeichen zwischen den Trennsymbolen, weil für jedes Zeichen in dem Teilwort u der Eingabe *höchstens ein* Zeichen hinter dem ersten Trennsymbol markiert wird. Die nicht eingezeichneten Transitionen stellen wiederum sicher, dass in einem solchen Fall die Funktion nicht definiert ist.
- In den Zuständen s, q_c, und q'_c wird sichergestellt, dass es genau zwei Trennzeichen gibt.
- Dadurch, dass die Turingmaschine zunächst markierte a-Symbole (also eigentlich $\underline{a}$-Symbole) zum zählen der Distanz anhängt, kann im Zustand q'_c sichergestellt werden, dass die Eingabe nicht die Form $u\$v\$x$ mit $|x| > 0$ hat. Im Zustand q'_c befindet sich der Zeiger nämlich hinter dem zweiten Trennsymbol und es können nur markierte Zeichen und das $\sqcup$-Symbol gelesen werden, ohne in den Zustand nein überzugehen. Mit anderen Worten: es können nur solche Symbole gelesen werden, die wir angehangen haben, denn markierte Symbole sind nicht im Eingabealphabet und die Turingmaschine markiert keine Symbole hinter dem zweiten Trennzeichen, die Teil der Eingabe waren.

  Tatsächlich wird dies auch schon, der gleichen Idee folgend, in dem Zustand q'_d

überprüft. Bis auf den Fall, dass die Eingabe von der Form $\$v\$x$ oder $w\$w\$x$ ist, denn in diesen Fällen wird die Maschine niemals in die Zustände q_d und q'_d übergehen.

b) Beobachtung: Ein Wort der Form $u\$v\$$ mit $|u| = |v|$ ist genau dann in L, wenn $\text{dist}(u, v) = 0$ gilt. Umgekehrt ist jedes Wort in L von der Form $u\$v\$$, wobei nicht nur $|u| = |v|$ sondern sogar $u = v$ gilt.

Für eine Turingmaschine, die genau alle Wörter in L akzeptieren soll, genügt es also zu prüfen, ob das Eingabewort von der Form $u\$v\$$ mit $|u| = |v|$ ist und $\text{dist}(u, v) = 0$ gilt. Da die Turingmaschine M, die wir in a) konstruiert haben, für Wörter, die *nicht* die Form $u\$v\$$ mit $|u| = |v|$ haben, nicht anhält oder in den Zustand nein übergeht, können wir sie wie folgt abwandeln, um L zu akzeptieren:

- Statt in den Haltezustand zu gehen, laufe wieder hinter das zweite Trennzeichen: steht dort ein a-Zeichen so gilt $\text{dist}(u, v) > 0$ und damit $u \neq v$. In diesem Fall wird in den Zustand nein übergegangen und damit verworfen.
- Andernfalls muss die Eingabe die Form $w\$w\$$ haben, sonst wäre die Distanz größer 0 oder M wäre nicht in den Haltezustand gewechselt. Die Maschine soll in diesem Fall in den Zustand ja übergehen, um zu akzeptieren.

Hinweis

Alternativ kann auch alle Transitionen, die in den Zustand q_d führen, stattdessen in den Zustand nein geführt werden (zur Erinnerung: im Zustand q_d haben wir unterschiedliche Symbole an der gleichen Position in u und v gefunden). Dann kann, statt in den Haltezustand, direkt in den Zustand ja übergegangen werden.