Prof. Dr. Jasmin Blanchette
Jannis Limperg
Lydia Kondylidou

Ludwig-Maximilians-Universität München
Institut für Informatik
Abgabe bis 28. Juni 2023, 10:00 Uhr

Lösungsvorschlag zur Übung 10 zur Vorlesung

Formale Sprachen und Komplexität

Erlaubte Konstrukte für

LOOP-Programme	WHILE-Programme	GOTO-Programme
$x_i := x_j + c$	$x_i := x_j + c$	$M_k : x_i := x_j + c$
$x_i := x_j - c$	$x_i := x_j - c$	$M_k : x_i := x_j - c$
$x_i := c$	$x_i := c$	$M_k : x_i := c$
$P_1; P_2$	$P_1; P_2$	$P_1; P_2$
LOOP x_i **DO** P **END**	**LOOP** x_i **DO** P **END**	M_k : **GOTO** M_j
	WHILE $x_i \neq 0$ **DO** P **END**	M_k : **IF** $x_i = c$ **THEN GOTO** M_j
		M_k : **HALT**

Dabei darf jede Marke M_k nur einmal vorkommen

FSK10-1 ***WHILE-, LOOP- und GOTO-Programme*** (2 Punkte)

In den folgenden Teilaufgaben sollen Sie jeweils ein Programm angeben. Verwenden Sie dabei nur die erlaubten Anweisungen aus der obigen Tabelle. Kommentieren Sie außerdem Ihre Programme, sodass klar wird, wie sie funktionieren sollen. Nicht kommentierte Programme werden eventuell nicht gewertet.

a) Schreiben Sie ein GOTO-Programm, das der Variablen x_0 den Wert von $x_1 + x_2$ zuweist.

LÖSUNGSVORSCHLAG:

// Initialisiere $x_0 = x_1$
M_1 : $x_0 := x_1 + 0$
// x_2-mal:
M_3 : **IF** $x_2 = 0$ **THEN GOTO** M_7
// Inkrementiere x_0
M_4 : $x_0 := x_0 + 1$
M_5 : $x_2 := x_2 - 1$
M_6 : **GOTO** M_3
M_7 : **HALT**

b) Schreiben Sie ein WHILE-Programm, das die folgende Funktion berechnet:

$$f : \mathbb{N}^2 \to \mathbb{N}, \quad f(x,y) = x * y + 1$$

LÖSUNGSVORSCHLAG:

Gemäß der Definition von WHILE-Berechenbarkeit erhalten wir die Eingaben x in x_1 und y in x_2 und speichern die Ausgabe in x_0.

```
// x-mal:
WHILE x_1 ≠ 0 DO
  // Addiere y zum Ergebnis:
  // Initialisiere x_3 = y
  x_3 := x_2;
  // y-mal:
  WHILE x_3 ≠ 0 DO
    // Inkrementiere das Ergebnis
    x_0 := x_0 + 1;
    x_3 := x_3 − 1
  END;
  x_1 := x_1 − 1
END;
// Addiere 1 zum Ergebnis
x_0 := x_0 + 1
```

c) Schreiben Sie ein LOOP-Programm, das den folgenden Pseudocode implementiert:

IF $x_1 \leq x_2$ **THEN** $x_0 := x_2 - x_1$ **ELSE** $x_0 := x_1 - x_2$ **END**

LÖSUNGSVORSCHLAG:

Programm	Effekt wenn $x_1 \leq x_2$	Effekt wenn $x_1 > x_2$
$x_3 := x_1 + 0$;	$x_3 = x_1$	$x_3 = x_1$
LOOP x_2 **DO** $x_3 := x_3 - 1$ **END**;	$x_3 = 0$	$x_3 = x_1 - x_2 \neq 0$
$x_4 := 1$;	$x_4 = 1$	$x_4 = 1$
$x_5 := 0$;	$x_5 = 0$	$x_5 = 0$
LOOP x_3 **DO** $x_4 := 0$ **END**;	—	$x_4 = 0$
LOOP x_3 **DO** $x_5 := 1$ **END**;	—	$x_5 = 1$
LOOP x_4 **DO** $x_0 := x_2 - x_1$ **END**;	$x_0 = x_2 - x_1$	—
LOOP x_5 **DO** $x_0 := x_1 - x_2$ **END**	—	$x_0 = x_1 - x_2$

Die Anweisungen der Form $x_i := x_j - x_k$ in den letzten zwei Zeilen sind Pseudocode, den wir noch in LOOP-Code übersetzen. In der vorletzten Zeile erhalten wir dann das Teilprogramm

LOOP x_4 **DO**
  LOOP x_1 **DO**
    $x_0 := x_2 - 1$
  END
END;

und entsprechend für die letzte Zeile.

FSK10-2 ***LOOPY-Programme*** (2 Punkte)

Wir betrachten LOOPY-Programme, die aus folgenden Anweisungen bestehen:

- $x_i := x_j + c$, $x_i := x_j - c$ und P_1; P_2 wie bei LOOP-Programmen.
- $(P_1 \mid P_2)$, wobei P_1 und P_2 LOOPY-Programme sind. Diese Anweisung führt nichtdeterministisch entweder P_1 oder P_2 aus.
- **LOOPY** P **END**. Diese Anweisung führt das LOOPY-Programm P nichtdeterministisch 0 oder mehr Male aus.

Beispielsweise führt das LOOPY-Programm

$$\textbf{LOOPY}\ (x_0 := x_0 + 2 \mid x_0 := x_0 - 1)\ \textbf{END}$$

beliebig oft entweder die Anweisung $x_0 := x_0 + 2$ oder die Anweisung $x_0 := x_0 - 1$ aus, wobei es in jeder Iteration eine andere Entscheidung treffen kann. Der Wert von x_0 am Ende des Programms kann also jede natürliche Zahl sein.

a) Die Semantik eines LOOPY-Programms können wir als eine Relation $\xrightarrow[\text{LOOPY}]{}$ definieren. Diese Relation ist ähnlich der für WHILE-Programme, aber LOOPY-Programme können nichtdeterministisch verschiedene Ergebnisse haben. Dementsprechend kann ein LOOPY-Programm P mit einer Variablenbelegung ρ mehrere Nachfolge-Variablenbelegungen ρ' und mehrere Nachfolge-Programme P' haben, je nachdem, welche nichtdeterministischen Entscheidungen das Programm trifft. Für alle diese ρ' und P' soll gelten:

$$(\rho, P) \xrightarrow[\text{LOOPY}]{} (\rho', P')$$

Definieren Sie die Semantik von LOOPY-Programmen.

LÖSUNGSVORSCHLAG:

Zuweisungen zu Variablen funktionieren genau so wie bei LOOP-Programmen:

$$(\rho, x_i := x_j + c) \xrightarrow[\text{LOOPY}]{} (\rho\{x_i \mapsto \rho(x_j) + c\}, \varepsilon)$$
$$(\rho, x_i := x_j - c) \xrightarrow[\text{LOOPY}]{} (\rho\{x_i \mapsto \max(0, \rho(x_j) - c)\}, \varepsilon)$$

Sequenzierung ebenfalls wie bei LOOP-Programmen. Wir wählen eine etwas andere Formulierung als im Skript, aber beide Formulierungen sind äquivalent.

$$(\rho, P_1;\ P_2) \xrightarrow[\text{LOOPY}]{} (\rho', P_2) \quad \text{falls } (\rho, P_1) \xrightarrow[\text{LOOPY}]{}{}^{*} (\rho', \varepsilon)$$

Für die nichtdeterministische Ausführung zweier Programme brauchen wir eine Regel für jede Alternative:

$$(\rho, (P_1 \mid P_2)) \xrightarrow[\text{LOOPY}]{} (\rho, P_1)$$
$$(\rho, (P_1 \mid P_2)) \xrightarrow[\text{LOOPY}]{} (\rho, P_2)$$

LOOPY führt das gegebene Programm beliebig oft aus. Wir kodieren das hier mit zwei Regeln, eine für „0-mal“ und eine für „$(n+1)$-mal“:

$$(\rho, \textbf{LOOPY } P \textbf{ END}) \xrightarrow[\text{LOOPY}]{} (\rho, \varepsilon)$$
$$(\rho, \textbf{LOOPY } P \textbf{ END}) \xrightarrow[\text{LOOPY}]{} (\rho, P;\ \textbf{LOOPY } P \textbf{ END})$$

b) Zeigen Sie, dass mit Ihrer Semantik gilt:

$$(\{x_0 \mapsto 0\}, \textbf{LOOPY } (x_0 := x_0 + 2 \mid x_0 := x_0 - 1) \textbf{ END}) \xrightarrow[\text{LOOPY}]{}{}^{*} (\{x_0 \mapsto 1\}, \varepsilon)$$

LÖSUNGSVORSCHLAG:

$(\{x_0 \mapsto 0\}, \textbf{LOOPY } (x_0 := x_0 + 2 \mid x_0 := x_0 - 1) \textbf{ END}) \xrightarrow[\text{LOOPY}]{}$
$(\{x_0 \mapsto 0\},$
$\quad (x_0 := x_0 + 2 \mid x_0 := x_0 - 1);\ \textbf{LOOPY } (x_0 := x_0 + 2 \mid x_0 := x_0 - 1) \textbf{ END}) \xrightarrow[\text{LOOPY}]{(1)}$
$(\{x_0 \mapsto 2\}, \textbf{LOOPY } (x_0 := x_0 + 2 \mid x_0 := x_0 - 1) \textbf{ END}) \xrightarrow[\text{LOOPY}]{}$
$(\{x_0 \mapsto 2\},$

$(x_0 := x_0 + 2 \mid x_0 := x_0 - 1);$ **LOOPY** $(x_0 := x_0 + 2 \mid x_0 := x_0 - 1)$ **END**) $\xrightarrow[\text{LOOPY}]{(2)}$
$(\{x_0 \mapsto 1\},$ **LOOPY** $(x_0 := x_0 + 2 \mid x_0 := x_0 - 1)$ **END**) $\xrightarrow[\text{LOOPY}]{}$
$(\{x_0 \mapsto 1\}, \varepsilon)$

Aussage (1) gilt, da

$$(\{x_0 \mapsto 0\}, (x_0 := x_0 + 2 \mid x_0 := x_0 - 1)) \xrightarrow[\text{LOOPY}]{}$$
$$(\{x_0 \mapsto 0\}, x_0 := x_0 + 2) \xrightarrow[\text{LOOPY}]{} (\{x_0 \mapsto 2\}, \varepsilon)$$

Analog für Aussage (2):

$$(\{x_0 \mapsto 2\}, (x_0 := x_0 + 2 \mid x_0 := x_0 - 1)) \xrightarrow[\text{LOOPY}]{}$$
$$(\{x_0 \mapsto 2\}, x_0 := x_0 - 1) \xrightarrow[\text{LOOPY}]{} (\{x_0 \mapsto 1\}, \varepsilon)$$

c) Eine Funktion $f\colon \mathbb{N}^k \to \mathbb{N}$ ist LOOPY-berechenbar, wenn es ein LOOPY-Programm P gibt sodass es für alle $n_1, \ldots, n_k \in \mathbb{N}$ eine Variablenbelegung ρ gibt, für die gilt:

$$(\{x_1 \mapsto n_1, \ldots, x_k \mapsto n_k\}, P) \xrightarrow[\text{LOOPY}]{}{}^{*} (\rho, \varepsilon)$$

und $\rho(x_0) = f(n_1, \ldots, n_k)$.

Zeigen Sie: jede Funktion, die LOOP-berechenbar ist, ist auch LOOPY-berechenbar.

LÖSUNGSVORSCHLAG:

Angenommen $f\colon \mathbb{N}^k \to \mathbb{N}$ ist LOOP-berechenbar, d.h. es gibt ein LOOP-Programm P sodass

$$(\{x_1 \mapsto n_1, \ldots, x_k \mapsto n_k\}, P) \xrightarrow[\text{LOOP}]{}{}^{*} (\rho, \varepsilon)$$

mit $\rho(x_0) = f(n_1, \ldots, n_k)$. Wir zeigen, dass es dann auch ein LOOPY-Programm P' gibt, für das gilt:

$$(\{x_1 \mapsto n_1, \ldots, x_k \mapsto n_k\}, P') \xrightarrow[\text{LOOPY}]{}{}^{*} (\rho, \varepsilon)$$

Daraus folgt direkt, dass f LOOPY-berechenbar ist.

Um die Behauptung zu zeigen, geben wir für jedes mögliche LOOP-Programm ein entsprechendes LOOPY-Programm an. Für dieses LOOPY-Programm muss

gelten, dass eine seiner möglichen (nichtdeterministischen) Ausführungen der Ausführung des LOOP-Programms entspricht.

Für die LOOP-Programme $x_i := x_j + c$, $x_i := x_j - c$ und $P_1;\ P_2$ ist das trivial, da diese Programme auch LOOPY-Programme sind.

Für das LOOP-Programm **LOOP** x_i **DO** P **END** wählen wir das LOOPY-Programm **LOOPY** P **END**. Wenn wir das LOOP-Programm ausführen, erhalten wir

$$(\rho, \textbf{LOOP } x_i \textbf{ DO } P \textbf{ END}) \xrightarrow[\text{LOOP}]{} (\rho, \underbrace{P;\ \ldots;\ P}_{\rho(x_i)\text{-mal}})$$

Das LOOPY-Programm generiert die Ausführungen

$$(\rho, \textbf{LOOPY } P \textbf{ END}) \xrightarrow[\text{LOOPY}]{}{}^{*} (\rho, \underbrace{P;\ \ldots;\ P}_{k\text{-mal}})$$

für beliebige k und damit auch die Ausführung des LOOP-Programms. (Hier braucht das LOOPY-Programm mehrere $\xrightarrow[\text{LOOPY}]{}$-Schritte, um einen $\xrightarrow[\text{LOOP}]{}$-Schritt zu simulieren, aber das schadet nicht. Wir hätten die Semantik von **LOOPY** auch analog zu der von **LOOP** definieren können, um diese Diskrepanz zu vermeiden.)

FSK10-3 ***FOR-Programme*** (0 Punkte)

a) Wir betrachten FOR-Programme. Diese haben dieselbe Syntax wie LOOP-Programme ohne das **LOOP**-Konstrukt, dafür gibt es ein **FOR**-Konstrukt:

$$\textbf{FOR } x_i := 1 \textbf{ TO } x_k \textbf{ DO } P \textbf{ END}$$

wobei P ein FOR-Programm ist, welches die Variable x_i nicht neu setzt (d.h. $x_i := \ldots$ kommt in P nicht vor, aber lesende Zugriffe $x_j := x_i + c$ und $x_j := x_i - c$ (mit $j \neq i$) sind erlaubt). Die informelle Semantik der **FOR**-Schleife ist, dass sie das Programm P N-Mal durchläuft, wobei N der Wert von x_k vor Eintritt in die FOR-Schleife ist und im j-ten Durchlauf $x_i := j$ gilt. Nach Beenden der Durchläufe gilt $x_i := N$.

Definieren Sie die operationale Semantik der FOR-Programme formal (analog zur operationalen Semantik der LOOP-Programme, siehe Abschnitt 10.1.2 im Skript), indem Sie einen Berechnungsschritt $(\rho, P) \xrightarrow[\text{FOR}]{} (\rho', P')$ definieren, wobei ρ eine Variablenbelegung und P ein FOR-Programm ist. Dabei muss die Semantik der **FOR**-Schleife der obigen informellen Semantik entsprechen und alle anderen Konstrukte müssen dieselbe Semantik wie in LOOP-Programmen haben.

LÖSUNGSVORSCHLAG:

Die Semantik von FOR-Programmen kann als Abfolge von Berechnungsschritten $\underset{\text{FOR}}{\longrightarrow}$ auf Paaren (ρ, P) beschrieben werden, wobei ρ eine Variablenbelegung und P ein FOR-Programm oder ein „leeres“ Programm ε ist.

Die Definition eines Berechnungsschrittes $\underset{\text{FOR}}{\longrightarrow}$ für eine Variablenbelegung ρ und ein FOR-Programm ist:

- $(\rho, x_i := c) \underset{\text{FOR}}{\longrightarrow} (\rho', \varepsilon)$ wobei $\rho' = \rho\{x_i \mapsto c\}$
- $(\rho, x_i := x_j + c) \underset{\text{FOR}}{\longrightarrow} (\rho', \varepsilon)$ wobei $\rho' = \rho\{x_i \mapsto n\}$ und $n = \rho(x_j) + c$
- $(\rho, x_i := x_j - c) \underset{\text{FOR}}{\longrightarrow} (\rho', \varepsilon)$ wobei $\rho' = \rho\{x_i \mapsto n\}$ und $n = \max(0, \rho(x_j) - c)$
- $(\rho, P_1; P_2) \underset{\text{FOR}}{\longrightarrow} (\rho', P; P_2)$ wenn $(\rho, P_1) \underset{\text{FOR}}{\longrightarrow} (\rho', P)$ und $P \neq \varepsilon$
- $(\rho, P_1; P_2) \underset{\text{FOR}}{\longrightarrow} (\rho', P_2)$ wenn $(\rho, P_1) \underset{\text{FOR}}{\longrightarrow} (\rho', \varepsilon)$
- $(\rho, \textbf{FOR } x_i := 1 \textbf{ TO } x_k \textbf{ DO } P \textbf{ END}) \underset{\text{FOR}}{\longrightarrow} (\rho, x_i := 0; \underbrace{x_i := x_i + 1; P; \ldots; x_i := x_i + 1; P}_{\rho(x_k)\text{-mal}})$

b) Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt *FOR-berechenbar*, wenn es ein FOR-Programm P gibt, sodass für alle $n_1, \ldots, n_k \in \mathbb{N}$ und Variablenbelegungen $\rho = \{x_1 \mapsto n_1, \ldots, x_k \mapsto n_k\}$ gilt: $(\rho, P) \underset{\text{FOR}}{\longrightarrow}^* (\rho', \varepsilon)$ und $\rho'(x_0) = f(n_1, \ldots, n_k)$, wobei $\underset{\text{FOR}}{\longrightarrow}^*$ für 0 oder beliebig viele Berechnungsschritte steht und ε das leere Programm ist.

Zeigen Sie, dass jede FOR-berechenbare Funktionen auch LOOP-berechenbar ist. Hierfür müssen Sie zeigen, dass jedes FOR-Programm als LOOP-Programm kodiert werden kann, wobei die berechnete Funktion dieselbe ist.

LÖSUNGSVORSCHLAG:

Sei $f : \mathbb{N}^k \to \mathbb{N}$ FOR-berechenbar. Dann gibt es ein FOR-Programm P, was die genannten Bedingungen erfüllt. Wir zeigen, dass es auch ein LOOP-Programm $\psi(P)$ gibt, welches f berechnet. $\psi(P)$ entsteht aus P, indem alle Konstrukte unverändert übernommen werden, bis auf **FOR**-Schleifen: **FOR** $x_i := 1$ **TO** x_k **DO** P **END** wird übersetzt in

$x_i := 0;$
LOOP x_k **DO**
$x_i := x_i + 1;$
$\psi(P)$
END

Es genügt zu zeigen, dass gilt $(\rho, P) \xrightarrow[\text{FOR}]{}^{*} (\rho', \varepsilon) \implies (\rho, \psi(P)) \xrightarrow[\text{LOOP}]{}^{*} (\rho', \varepsilon)$. wobei ε das leere Programm ist, denn dann berechnet $\psi(P)$ dieselbe Funktion wie P.

Da die Berechnung sequentiell und eindeutig ist, genügt es zu zeigen, dass gilt: $(\rho, P) \xrightarrow[\text{FOR}]{}^{+} (\rho', P') \implies (\rho, \psi(P)) \xrightarrow[\text{LOOP}]{}^{+} (\rho', \psi(P'))$

Dabei bedeutet $\xrightarrow[\text{FOR}]{}^{+}, \xrightarrow[\text{LOOP}]{}^{+}$, dass mindestens ein Reduktionsschritt gemacht werden muss.

Das ist für alle Konstrukte bis auf **FOR**-Schleifen trivial und für **FOR**-Schleifen gilt:

$$
\begin{aligned}
& (\rho, \textbf{FOR } x_i := 1 \textbf{ TO } x_k \textbf{ DO } P \textbf{ END}) \\
\xrightarrow[\text{FOR}]{} \quad & (\rho, x_i := 0; \underbrace{x_i := x_i + 1; P; \ldots; x_i := x_i + 1; P}_{\rho(x_k)\text{-mal}}) \\
\xrightarrow[\text{FOR}]{} \quad & (\rho\{x_i \mapsto 0\}, \underbrace{x_i := x_i + 1; P; \ldots; x_i := x_i + 1; P}_{\rho(x_k)\text{-mal}})
\end{aligned}
$$

und

$$
\begin{aligned}
& (\rho, \psi(\textbf{FOR } x_i := 1 \textbf{ TO } x_k \textbf{ DO } P \textbf{ END})) \\
= \quad & (\rho, x_i := 0; \textbf{LOOP } x_k \textbf{ DO } x_i := x_i + 1; \psi(P) \textbf{ END}) \\
\xrightarrow[\text{LOOP}]{} \quad & (\rho\{x_i \mapsto 0\}, \textbf{LOOP } x_k \textbf{ DO } x_i := x_i + 1; \psi(P) \textbf{ END}) \\
\xrightarrow[\text{LOOP}]{} \quad & (\rho\{x_i \mapsto 0\}, \underbrace{x_i := x_i + 1; \psi(P); \ldots; x_i := x_i + 1; \psi(P)}_{\rho(x_k)\text{-mal}}) \\
= \quad & (\rho\{x_i \mapsto 0\}, \psi(\underbrace{x_i := x_i + 1; P; \ldots; x_i := x_i + 1; P}_{\rho(x_k)\text{-mal}}))
\end{aligned}
$$

Daher gilt $(\rho, \psi(P)) \xrightarrow[\text{LOOP}]{}^{*} (\rho', \varepsilon)$ und $\psi(P)$ berechnet f.

c) Zeigen sie, dass jede LOOP-berechenbare Funktion auch FOR-berechenbar ist. Hierfür müssen Sie zeigen, dass jedes LOOP-Programm als FOR-Programm kodiert werden kann, wobei die berechnete Funktion dieselbe ist.

LÖSUNGSVORSCHLAG:

Sei $f\colon \mathbb{N}^k \to \mathbb{N}$ eine LOOP-berechenbare Funktion. Dann gibt es ein LOOP-Programm P, welches f berechnet. Sei ϕ die Übersetzungsfunktion von LOOP-Programmen in FOR-Programme, die alle Konstrukte unverändert lässt und **LOOP** x_i **DO** P **END** übersetzt in **FOR** $x_n := 1$ **TO** x_i **DO** $\phi(P)$ **END**, wobei x_n eine neue Variable ist.

Sei $(\rho, P) \xrightarrow[\text{LOOP}]{}^{*} (\rho', \varepsilon)$ mit $\rho'(x_0) = f(\rho(x_1), \ldots, \rho(x_k))$.

Alle $\xrightarrow[\text{LOOP}]{}$-Schritte sind nach der Übersetzung auch $\xrightarrow[\text{FOR}]{}$-Schritte (mit gleichem ρ) außer für **LOOP**. Dort gilt $(\rho, \textbf{LOOP } x_i \textbf{ DO } P \textbf{ END}) \xrightarrow[\text{LOOP}]{} (\rho, \underbrace{P; \ldots; P}_{\rho(x_k)\text{-mal}})$ und

$$\begin{array}{ll} & (\rho, \phi(\textbf{LOOP } x_i \textbf{ DO } P \textbf{ END})) \\ = & (\rho, \textbf{FOR } x_n := 1 \textbf{ TO } x_i \textbf{ DO } \phi(P) \textbf{ END}) \\ \xrightarrow[\text{FOR}]{} & (\rho, x_n := 0; \underbrace{x_n := x_n + 1; \phi(P); \ldots, x_n := x_n + 1; \phi(P)}_{\rho(x_k)\text{-mal}}) \\ \xrightarrow[\text{FOR}]{} & (\rho\{x_n \to 0\}, \underbrace{x_n := x_n + 1; \phi(P); \ldots, x_n := x_n + 1; \phi(P)}_{\rho(x_k)\text{-mal}}) \end{array}$$

Da x_n nicht in P vorkommt, kommt x_n auch nicht in $\phi(P)$ vor. Daher hat $\rho(x_n)$ keinen Einfluss auf die Berechnung und auch nicht auf das Gesamtergebnis, d.h. $(\rho, \phi(P)) \xrightarrow[\text{LOOP}]{}{}^* (\rho'', \varepsilon)$ mit $\rho''(x_0) = \rho'(x_0)$. Daher berechnet $\phi(P)$ ebenfalls f.

FSK10-4 ***Turingmaschinen*** (0 Punkte)

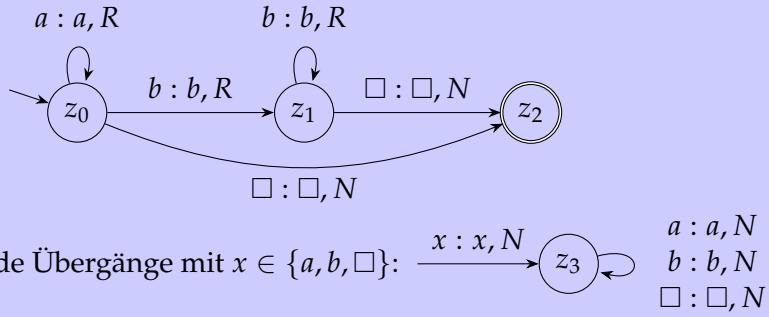
a) Seien $M = \{w \mid \#_a(w) = \#_b(w)\}$ und $N = L(a^*b^*)$ Sprachen über $\Sigma = \{a, b\}$. Zeigen Sie, dass es eine Funktion f gibt mit $\forall x \in \Sigma^* \,.\, x \in M \iff f(x) \in N$.

LÖSUNGSVORSCHLAG:

$$f(x) = \begin{cases} ab & \text{wenn} \quad \#_a(x) = \#_b(x) \\ ba & \text{wenn} \quad \#_a(x) \neq \#_b(x) \end{cases}$$

b) Geben Sie eine deterministische Turingmaschine an, welche N erkennt.

LÖSUNGSVORSCHLAG:



Fehlende Übergänge mit $x \in \{a, b, \square\}$:

c) Was muss für f gelten, damit man daraus folgern kann, dass M von einer Turingmaschine erkannt werden kann (ohne die Turingberechenbarkeit von M direkt zu zeigen)?

LÖSUNGSVORSCHLAG: f muss turingberechenbar sein. Sei F die passende Turingmaschine. Dann kann man M erkennen, indem man zunächst F ausführt und danach die Turingmaschine für N ausführt. Wie wir im Laufe der Vorlesung sehen, nennt man f in diesem Fall eine Reduktionsfunktion, für eine Reduktion von M auf N.