Berechenbarkeits- und Komplexitätstheorie

Lerneinheit 1: Turingmaschinen

Prof. Dr. Christoph Karg

Studiengang Informatik
Hochschule Aalen

Wintersemester 2015/2016

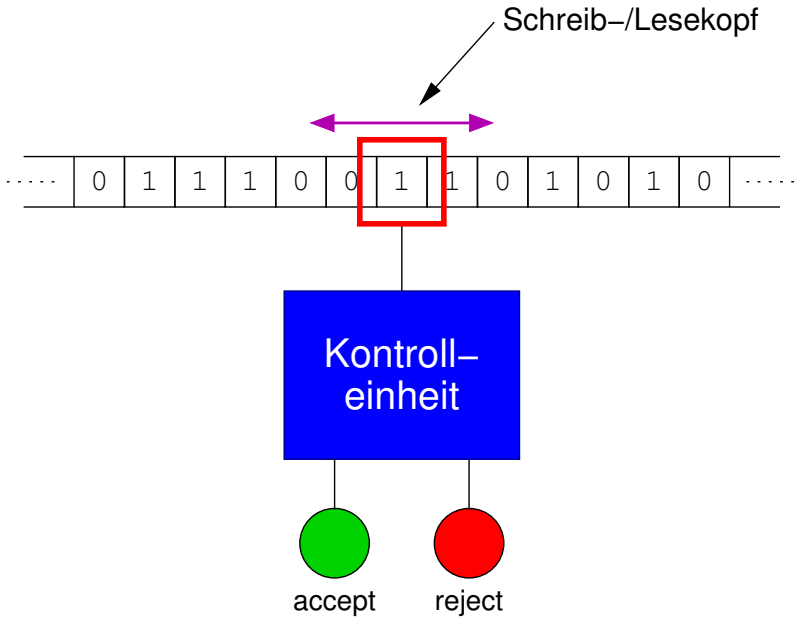
6.10.2015

Einleitung

Diese Lerneinheit beschäftigt sich mit Turingmaschinen. Ziel ist die Beantwortung der folgenden Fragen:

- Wie ist eine Turingmaschine aufgebaut?
- Wie arbeitet eine Turingmaschine?
- Was versteht man unter einer Konfiguration?
- Ist Nichtdeterminismus leistungsfähiger als Determinismus?

Aufbau einer Turingmaschine



Definition Turingmaschine

Definition. Eine Turingmaschine wird beschrieben durch ein Tupel $M = (Z, \Gamma, \Sigma, \sqcup, \delta, z_s, z_{acc}, z_{rej})$, wobei

- Z ist eine endliche Menge von Zuständen mit mindestens zwei Zuständen z_{acc} und z_{rej}
- Γ ist das Arbeitsalphabet
- $\Sigma \subseteq \Gamma$ ist das Eingabealphabet
- $\sqcup \in \Gamma - \Sigma$ ist das Blank-Symbol
- $\delta : Z \times \Gamma \mapsto Z \times \Gamma \times \{L, R, N\}$ ist die Überführungsfunktion
- z_{acc} und z_{rej} sind der akzeptierende bzw. verwerfende Endzustand
- z_s ist der Startzustand

Arbeitsweise

Betrachte die Turingmaschine $M = (Z, \Gamma, \Sigma, \sqcup, \delta, z_s, z_{acc}, z_{rej})$.

Verarbeitung der Eingabe $x = a_1 \dots a_n \in \Sigma^*$:

1. Initialisierung der Turingmaschine
   - Das Arbeitsband enthält die Sequenz

$\dots$	$\sqcup$	$\sqcup$	a_1	a_2	$\dots$	a_n	$\sqcup$	$\sqcup$	$\dots$

   - Der Schreib-/Lesekopf befindet sich über dem Buchstaben a_1
   - Der aktuelle Zustand ist der Startzustand z_s

Arbeitsweise (Forts.)

2. Schrittweise Berechnung unter Verwendung von δ sowie dem aktuellen Zustand z und dem Symbol a unter dem S/L-Kopf:
   - Falls $\delta(z, a) = (z', b, L)$, dann wechselt M in den Zustand z', ersetzt a durch b auf dem Arbeitsband und bewegt den S/L-Kopf nach links.
   - Falls $\delta(z, a) = (z', b, R)$, dann wechselt M in den Zustand z', ersetzt a durch b auf dem Arbeitsband und bewegt den S/L-Kopf nach rechts.
   - Falls $\delta(z, a) = (z', b, N)$, dann wechselt M in den Zustand z', ersetzt a durch b auf dem Arbeitsband und bewegt den S/L-Kopf nicht.
3. Wiederholung von Schritt 2, bis sich M im Zustand z_{acc} oder z_{rej} befindet.

Arbeitsweise (Forts.)

Ergebnis der Berechnung:

- Befindet sich M im Zustand z_{acc}, dann wird die Eingabe x akzeptiert.
- Befindet sich M im Zustand z_{rej}, dann wird die Eingabe x verworfen.
- Gelangt M während der Verarbeitung von x nie in einen der Zustände z_{acc} und z_{rej}, dann ist das Ergebnis unbestimmt.

Die von M akzeptierte Sprache ist

$$L(M) = \{x \in \Sigma^* \mid M \text{ akzeptiert } x\}.$$

Beispiel

Aufgabe: Konstruktion einer Turingmaschine, die die Sprache

$$L = \{0^{2^n} \mid n \geq 0\} = \{0, 00, 0000, 00000000, \ldots\}$$

erkennt.

Hinweis: Benutze die Formel

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$$

als Grundlage für den Algorithmus.

Beispiel (Forts.)

Algorithmus

1. Streiche die erste 0 der Eingabe (d.h. 0 durch X ersetzen).
2. Streiche die nächste 0.
3. Überlese die nächste 0 und streiche die darauf folgende 0. Falls dies nicht möglich ist, dann verwerfe.
4. Wiederhole Schritt 3, bis das Ende des Bandes erreicht ist.
5. Bewege den S/L-Kopf an das linke Ende des benutzten Teils des Arbeitsbandes.
6. Falls alle 0en gestrichen wurden, dann akzeptiere. Ansonsten springe zu Schritt 2.

Beispiel (Forts.)

Verarbeitung von $x = 0^8$:

- Inhalt des Arbeitsbands zu Beginn:

...	␣	0	0	0	0	0	0	0	0	␣	...

- Inhalt vor dem ersten Streichvorgang:

...	␣	X	0	0	0	0	0	0	0	␣	...

- Inhalt nach ersten Streichvorgang:

...	␣	X	X	0	X	0	X	0	X	␣	...

Beispiel (Forts.)

- Inhalt nach zweiten Streichvorgang:

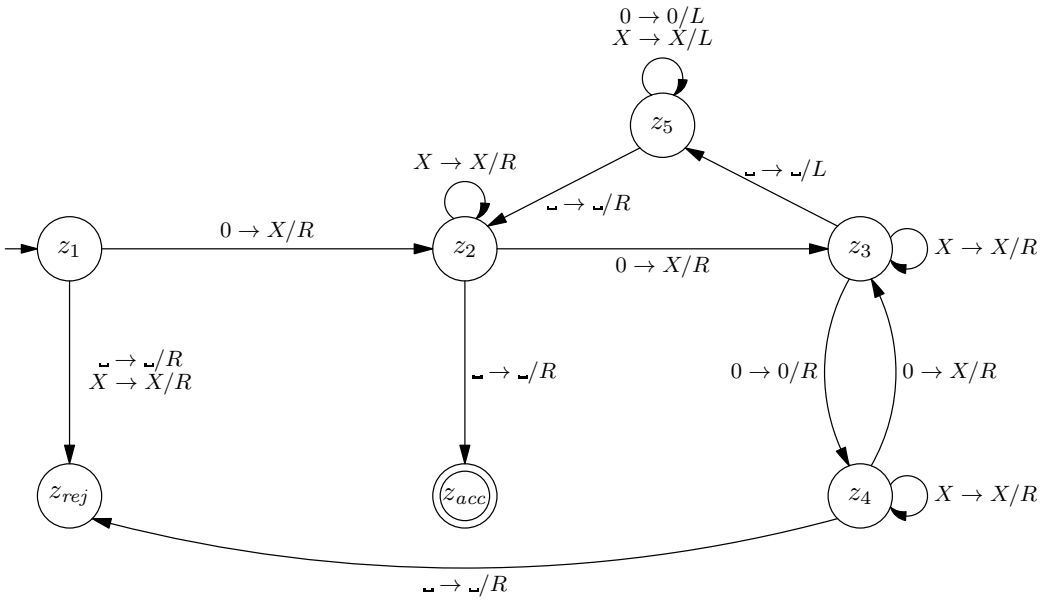
...	␣	X	X	X	X	0	X	X	X	␣	...

- Inhalt nach dritten Streichvorgang:

...	␣	X	X	X	X	X	X	X	X	␣	...

$\rightsquigarrow$ M akzeptiert das Wort 0^8

Beispiel (Forts.)



Beispiel (Forts.)

Verarbeitung von $x = 0^4 = 0000$:

$$
\begin{array}{ll}
 & z_1 0000 \\
\to_M & X z_2 000 \\
\to_M & XX z_3 00 \\
\to_M & XX0 z_4 0 \\
\to_M & XX0X z_3 \sqcup \\
\to_M & XX0 z_5 X \sqcup \\
\to_M & XX z_5 0X \sqcup \\
\to_M & X z_5 X0X \sqcup \\
\to_M & \sqcup z_5 XX0X \sqcup \\
\to_M & z_5 \sqcup XX0X \sqcup \\
\to_M & \sqcup z_2 XX0X \sqcup \\
\to_M & \sqcup X z_2 X0X \sqcup \\
\to_M & \sqcup XX z_2 0X \sqcup \\
\to_M & \sqcup XXX z_3 X \sqcup \\
\to_M & \sqcup XXXX z_3 \sqcup \\
\to_M & \sqcup XXX z_5 X \sqcup \\
\to_M & \sqcup XX z_5 XX \sqcup \\
\to_M & \sqcup X z_5 XXX \sqcup \\
\to_M & z_5 \sqcup XXXX \sqcup \\
\to_M & \sqcup z_2 XXXX \sqcup \\
\to_M & \sqcup X z_2 XXX \sqcup \\
\to_M & \sqcup XX z_2 XX \sqcup \\
\to_M & \sqcup XXX z_2 X \sqcup \\
\to_M & \sqcup XXXX z_2 \sqcup \\
\to_M & \sqcup XXXX \sqcup z_{acc}
\end{array}
$$

Beispiel (Forts.)

Frage: Arbeitet M korrekt?

Ansatz: Betrachte die Anzahl n der verbleibenden Nullen, wenn M den Zustand z_2 annimmt.

- Ist n gerade, dann verwirft M.
- Ist n ungerade, dann führt M einen kompletten Streichvorgang. durch. Danach gilt:
  - Die Zahl der Xe erhöht sich um $\left\lceil \frac{n}{2} \right\rceil$.
  - Die Zahl der verbleibenden Nullen ist $\left\lfloor \frac{n}{2} \right\rfloor$.

Fall 1: Eingabe 0^n, wobei n ungerade. Nach dem Streichen der ersten Null, ist die Anzahl der verbleibenden Nullen gerade.

$\rightsquigarrow$ M verwirft die Eingabe.

Beispiel (Forts.)

Fall 2: Eingabe 0^{2^k}, wobei $k \geq 0$.

Beachte: $\left\lceil \frac{2^\ell - 1}{2} \right\rceil = 2^{\ell-1}$, $\left\lfloor \frac{2^\ell - 1}{2} \right\rfloor = 2^{\ell-1} - 1$

Durchlauf	Anzahl Xe	Anzahl 0en
–	1	$2^k - 1$
1	$1 + 2^{k-1}$	$2^{k-1} - 1$
2	$1 + 2^{k-1} + 2^{k-2}$	$2^{k-2} - 1$
3	$1 + 2^{k-1} + 2^{k-2} + 2^{k-3}$	$2^{k-3} - 1$
$\vdots$	$\vdots$	$\vdots$
$k-1$	$1 + 2^{k-1} + 2^{k-2} + \ldots + 2^2 + 2^1$	$2^1 - 1$
k	$1 + 2^{k-1} + 2^{k-2} + \ldots + 2^2 + 2^1 + 2^0$	0

$\rightsquigarrow$ M akzeptiert die Eingabe.

Beispiel (Forts.)

Fall 3: Eingabe $0^{2^k z}$, wobei $k \geq 1$ und $z > 1$ ungerade

Durchlauf	Anzahl X	Anzahl 0
–	1	$2^k z - 1$
1	$1 + 2^{k-1} z$	$2^{k-1} \cdot z - 1$
2	$1 + 2^{k-1} z + 2^{k-2} z$	$2^{k-2} \cdot z - 1$
3	$1 + 2^{k-1} z + 2^{k-2} z + 2^{k-3} z$	$2^{k-3} \cdot z - 1$
$\vdots$	$\vdots$	$\vdots$
$k-1$	$1 + 2^{k-1} z + 2^{k-2} z + \ldots + 2^2 z + 2^1 z$	$2^1 \cdot z - 1$
k	$1 + 2^{k-1} z + 2^{k-2} z + \ldots + 2^1 z + 2^0 z$	$2^0 \cdot z - 1$

$\rightsquigarrow$ M verwirft die Eingabe, da die Anzahl verbleibender 0en gerade ist.

Fazit: $L(M) = \{0^{2^k} \mid k \geq 0\}$.

Konfigurationen

Aufgabe: Beschreibung des Gesamtzustands der Turingmaschine, die eine Fortsetzung der Berechnung ermöglicht ohne die vorangegangenen Rechenschritte zu kennen.

Lösung: Einsatz von Konfigurationen.

Definition. Eine Konfiguration ist eine Momentaufnahme einer Turingmaschine während der Verarbeitung einer Eingabe. Sie beinhaltet folgende Informationen:

- Zustand der Turingmaschine
- Position des Schreib-/Lesekopfs
- Inhalt des beschriebenen Teils des Arbeitsbandes

Konfigurationen (Forts.)

Betrachte die Turingmaschine $M = (Z, \Gamma, \Sigma, \sqcup, \delta, z_{acc}, z_{rej})$.

Ein Konfiguration von M ist ein Wort

$$C \in \Gamma^* \times Z \times \Gamma^+.$$

Interpretation der Konfiguration $C = (u, z, v)$:

- Der Inhalt des Arbeitsbandes ist $\ldots \sqcup\sqcup uv \sqcup\sqcup \ldots$.
- M befindet sich im Zustand z.
- Der Schreib-/Lesekopf von M befindet sich über dem ersten Buchstaben von v.

Konfigurationen (Forts.)

Besondere Konfigurationen:

- Startkonfiguration von M auf Eingabe $x \in \Sigma^+$: (ε, z_s, x)
- Startkonfiguration von M auf Eingabe ε: $(\varepsilon, z_s, \sqcup)$.
- Akzeptierende Konfiguration: (u, z_{acc}, v), wobei $u \in \Gamma^*$ und $v \in \Gamma^+$.
- Verwerfende Konfiguration: (u, z_{rej}, v), wobei $u \in \Gamma^*$ und $v \in \Gamma^+$.

Einsatz von Konfigurationen

Betrachte die Konfiguration $C = (ua, z, bv)$, wobei $u, v \in \Gamma^*$, $a, b \in \Gamma$ und $z \in Z$

- Falls $\delta(z, b) = (z', c, L)$, dann ist $C' = (u, z', acv)$ eine Folgekonfiguration von C.
- Falls $\delta(z, b) = (z', c, R)$, dann ist $C' = (uac, z', v)$ eine Folgekonfiguration von C.
- Falls $\delta(z, b) = (z', c, N)$, dann ist $C' = (ua, z', cv)$ eine Folgekonfiguration von C.

Kurzschreibweise: $C \rightarrow_M C'$

Einsatz von Konfigurationen (Forts.)

Die Turingmaschine M akzeptiert die Eingabe x, genau dann wenn es eine Folge von Konfigurationen $C_1, C_2, \ldots, C_k$ gibt mit:

- $C_1 = (\varepsilon, z_s, x)$ ist die Startkonfiguration von M auf Eingabe x.
- Für alle $i = 2, \ldots, k$ gilt: $C_{i-1} \rightarrow_M C_i$
- C_k ist eine akzeptierende Konfiguration.

Analog: für das Verwerfen einer Eingabe.

Achtung: Es gibt Turingmaschinen, die bei bestimmten Eingaben nicht stoppen.

Church-Turing These

Church-Turing These:

> Die Menge der algorithmisch lösbaren Probleme ist identisch mit der durch Turingmaschinen berechenbare Probleme.

Konsequenz:

- "Programmierung" von Turingmaschinen mittels Pseudo-Kode.
- Einsatz von Turingmaschinen in Beweisen.

Mehrband-Turingmaschinen

Modifikation: Statte die Turingmaschine mit $k \geq 2$ Arbeitsbändern aus.

Formal: Überführungsfunktion

$$\delta : Z \times \Gamma^k \mapsto Z \times \Gamma^k \times \{L, R, N\}^k$$

Vereinbarung: Die Eingabe steht auf dem ersten Arbeitsband.

Arbeitsweise: Analog zur Einband-Turingmaschine.

Beispiel

Aufgabe: Konstruktion einer Mehrband-Turingmaschine, die die Sprache

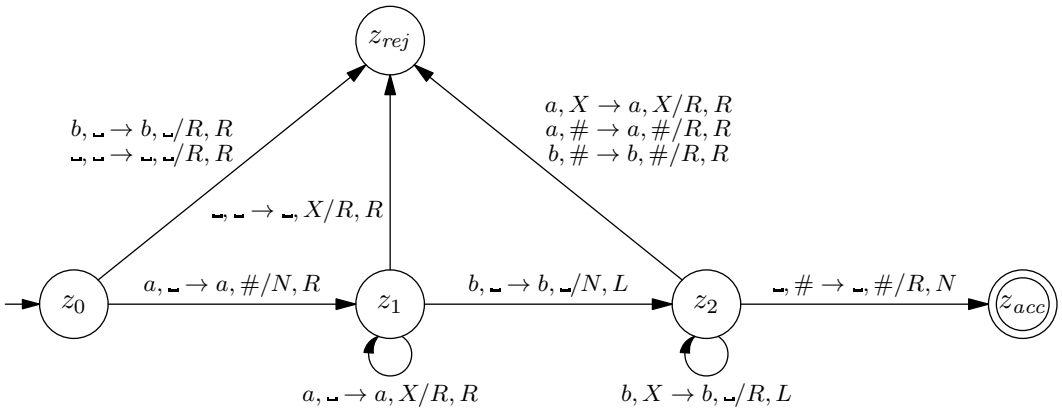
$$L = \{a^n b^n \mid n \geq 1\}$$

akzeptiert.

Idee: Benutze zwei Arbeitsbänder.

- Band 1 $\rightsquigarrow$ Eingabe
- Band 2 $\rightsquigarrow$ Keller

Beispiel (Forts.)



Beachte: Nicht eingezeichnete Übergänge führen zum verwerfenden Endzustand z_{rej}.

Ein Band genügt!

Satz. Jede durch eine Mehrband-Turingmaschine akzeptierbare Sprache ist durch eine Einband-Turingmaschine akzeptierbar.

Beweis. Betrachte eine k-Band-Turingmaschine M mit dem Arbeitsalphabet Γ, dem Eingabealphabet Σ und der Überführungsfunktion δ.

Ziel: Konstruktion einer Einband-Turingmaschine M' mit $L(M') = L(M)$.

Idee: Schreibe den Inhalt der k Arbeitsbänder hintereinander auf das Arbeitsband.

Ein Band genügt! (Forts.)

Neue Symbole: Neben Γ enthält das Arbeitsalphabet Γ' von M' die folgenden neuen Symbole:

- Position des S/L-Kopfes: $\dot{a}$ für alle $a \in \Gamma$.
- Bandgrenzen: $\triangleright$, $\triangleleft$.

Aufteilung des Arbeitsbands:

$$\underbrace{\triangleright\sqcup a_1^1 \dots \dot{a}_i^1 \dots a_{n_1}^1 \sqcup\triangleleft}_{\text{Band 1}}\underbrace{\triangleright\sqcup a_1^2 \dots \dot{a}_i^2 \dots a_{n_2}^2 \sqcup\triangleleft}_{\text{Band 2}}\dots\underbrace{\triangleright\sqcup a_1^k \dots \dot{a}_i^k \dots a_{n_k}^k \sqcup\triangleleft}_{\text{Band } k}$$

Ein Band genügt! (Forts.)

Algorithmus von M' zur Verarbeitung von $x = a_1 \dots a_n$:

1. Ersetze $a_1 \dots a_n$ auf dem Arbeitsband durch

$$\triangleright\sqcup\dot{a}_1 a_2 \dots a_n \sqcup\triangleleft\triangleright\dot{\sqcup}\sqcup\triangleleft\triangleright\sqcup\dot{\sqcup}\triangleleft\dots\triangleright\sqcup\dot{\sqcup}\triangleleft$$

2. Scanne das Band von links nach rechts und ermittle den Vektor $(b_1, \dots, b_k)$ der Symbole unter den jeweiligen S/L-Köpfen.
3. Berechne $(z', c_1, \dots, c_k, D_1, \dots, D_k) = \delta(z, b_1, \dots, b_k)$.
4. Falls $z' = z_{acc}$, dann akzeptiere. Falls $z' = z_{rej}$, dann verwerfe. Ansonsten ändere die Arbeitsbänder entsprechend ab.
5. Falls eine der Positionen der S/L-Köpfe an den Rand angrenzt, dann füge eine neue Bandzelle zum Bandsegment hinzu.
6. Springe zu Schritt 2.

Nichtdeterministische Turingmaschinen

Definition. Eine nichtdeterministische Turingmaschine wird beschrieben durch ein Tupel $M = (Z, \Gamma, \Sigma, \sqcup, \delta, z_s, z_{acc}, z_{rej})$, wobei

- Z ist eine endliche Menge von Zuständen mit mindestens zwei Zuständen z_{acc} und z_{rej}
- Γ ist das Arbeitsalphabet
- $\Sigma \subseteq \Gamma$ ist das Eingabealphabet
- $\sqcup \in \Gamma - \Sigma$ ist das Blank-Symbol
- $\delta : Z \times \Gamma \mapsto \mathcal{P}(Z \times \Gamma \times \{L, R\})$ ist die Überführungsfunktion
- z_{acc} und z_{rej} sind der akzeptierende bzw. verwerfende Endzustand
- z_s ist der Startzustand

Bemerkungen

- Nichtdeterminismus ist eine Art von paralleler Datenverarbeitung, mit der man effizient eine Vielzahl von Alternativen analysieren kann.
- Eine nichtdeterministische Turingmaschine akzeptiert eine Eingabe, wenn mindestens eine der Berechnungen in einer akzeptierenden Konfiguration endet.
- Die Arbeitsweise einer nichtdeterministischen Turingmaschine wird in Form eines Berechnungsbaums dargestellt.

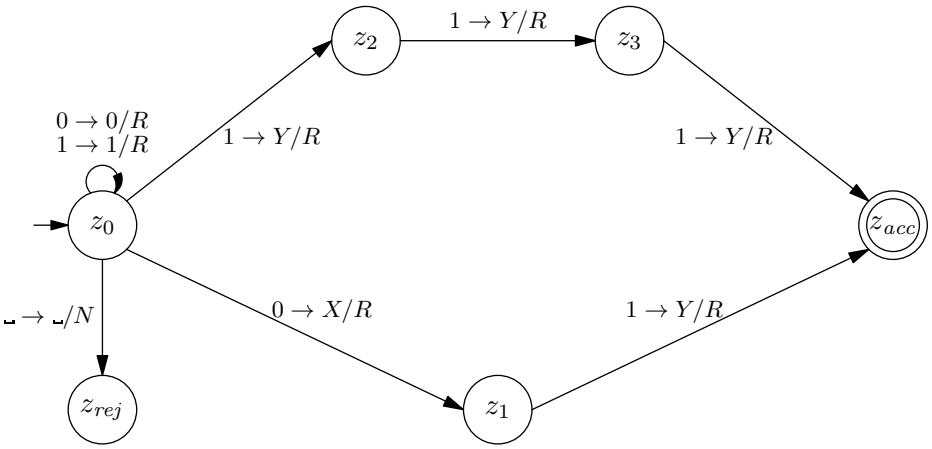
Beispiel

Aufgabe: Konstruktion einer nichtdeterministischen Turingmaschine, die die Sprache

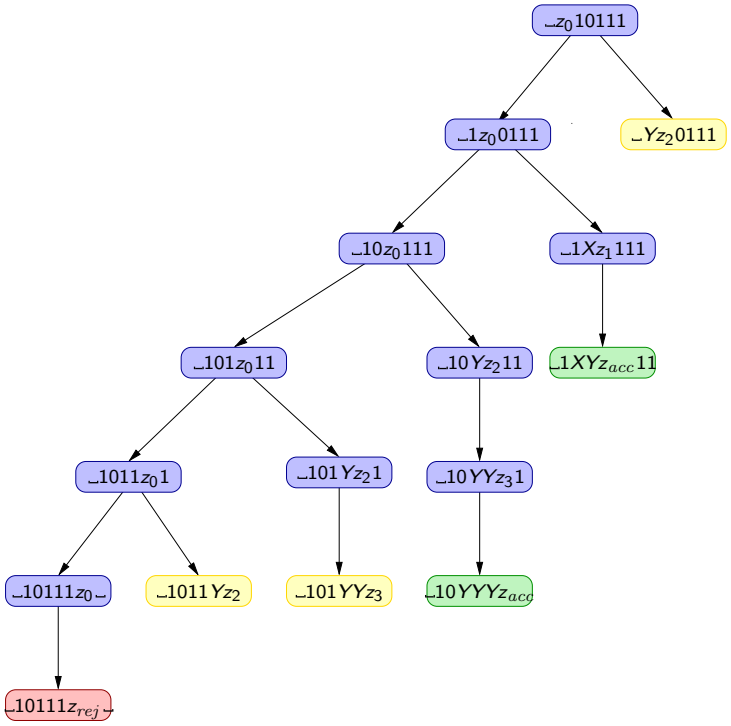
$$L = \{x \in \{0,1\}^* \mid x \text{ enthält 01 oder 111}\}$$

Idee: Parallele Suche nach einem der beiden Pattern.

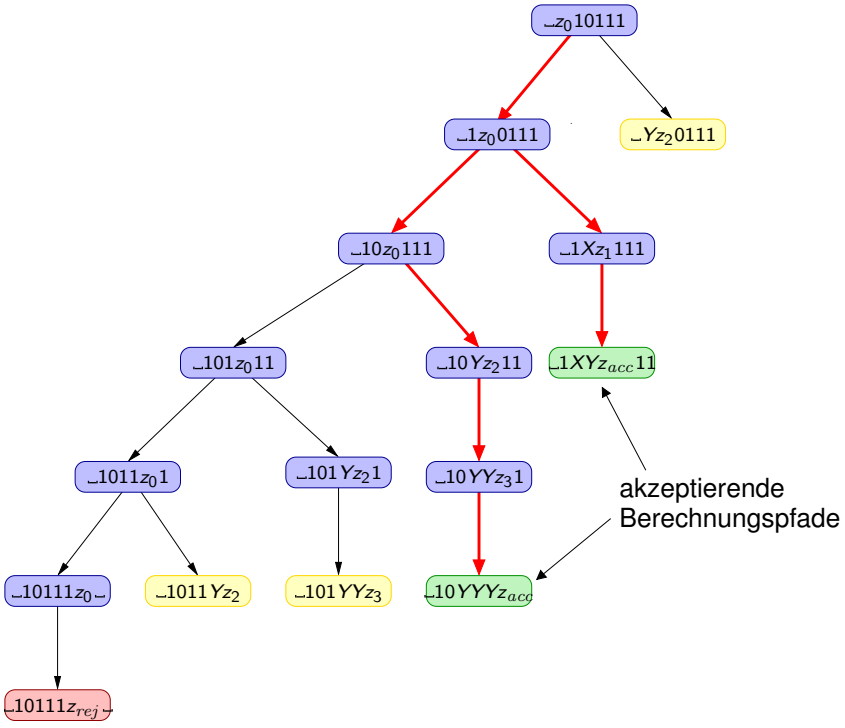
Beispiel (Forts.)



Beispiel (Forts.)



Beispiel (Forts.)



Ein wichtiger Satz

Satz. Jede durch eine nichtdeterministische Turingmaschine akzeptierbare Sprache ist durch eine deterministische Turingmaschine akzeptierbar.

Beweis. Betrachte die nichtdeterministische Turingmaschine $M = (Z, \Gamma, \Sigma, \sqcup, \delta, z_s, z_{acc}, z_{rej})$ und das Wort $x \in \Sigma^*$.

Idee: Durchsuche den Berechnungsbaum von M auf Eingabe x nach einer akzeptierenden Berechnung.

Problem: es existieren vielleicht unendlich lange Berechnungspfade.

Ein wichtiger Satz (Forts.)

$\textsc{SearchTree}_M(x)$
Input: Wort $x = a_1 \dots a_n \in \Sigma^*$
Output: `true`, falls M das Wort x akzeptiert, `false` sonst.

```
1 L := [(⊔, z_s, a_1 ... a_n)]
2 while L ≠ ∅ do
3   C := HEAD(L)
4   for jede Folgekonfiguration C' von C do
5     if C' ist eine akzeptierende Konfiguration then
6       return true
7     else if C' ist keine verwerfende Konfiguration then
8       Hänge C' an das Ende von L an
9 return false
```

Ein wichtiger Satz (Forts.)

Bemerkungen:

- $\mathcal{L}$ ist eine verkettete Liste, in der Konfigurationen gespeichert werden.
- Der Algorithmus führt eine Breitensuche im Berechnungsbaum von M auf Eingabe x durch.
- Falls M die Eingabe x akzeptiert, dann ist $\textsc{SearchTree}_M(x) = \texttt{true}$
- Falls M die Eingabe x auf allen Berechnungspfaden verwirft, dann ist $\textsc{SearchTree}_M(x) = \texttt{false}$.
- Falls M die Eingabe x nicht akzeptiert und es unendlich lange Berechnungspfade gibt, dann läuft $\textsc{SearchTree}_M(x)$ unendlich lange.

Zurück zum Beispiel

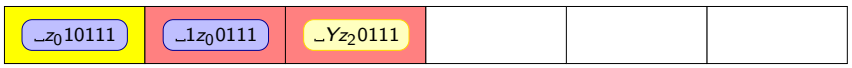
FIFO–Queue:

$\llcorner z_0 10111$					

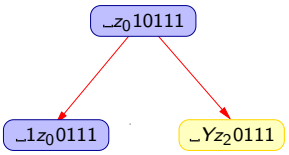
Ausgewerteter Teil des Berechnungsbaums:

Zurück zum Beispiel (Forts.)

FIFO–Queue:

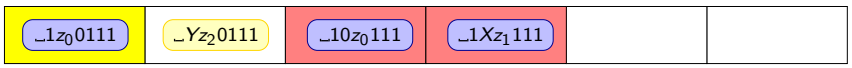


Ausgewerteter Teil des Berechnungsbaums:

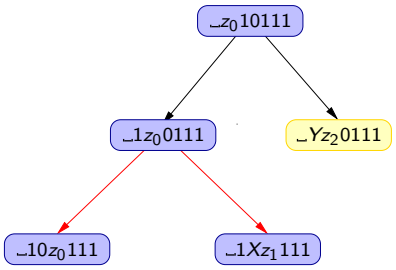


Zurück zum Beispiel (Forts.)

FIFO–Queue:

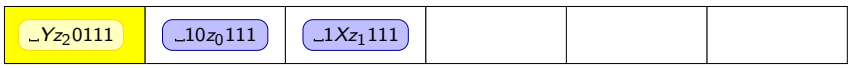


Ausgewerteter Teil des Berechnungsbaums:

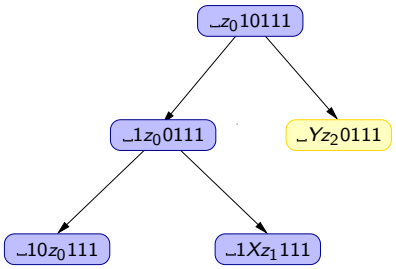


Zurück zum Beispiel (Forts.)

FIFO–Queue:

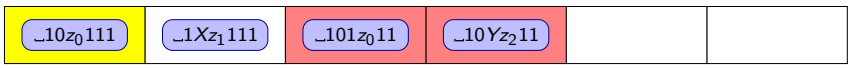


Ausgewerteter Teil des Berechnungsbaums:

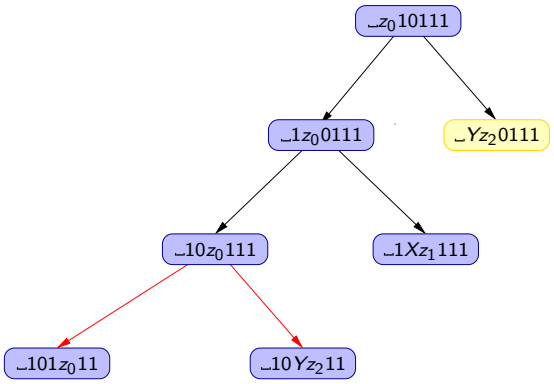


Zurück zum Beispiel (Forts.)

FIFO–Queue:

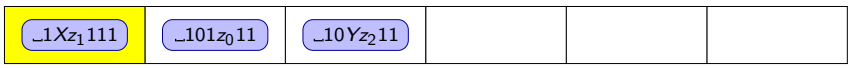


Ausgewerteter Teil des Berechnungsbaums:

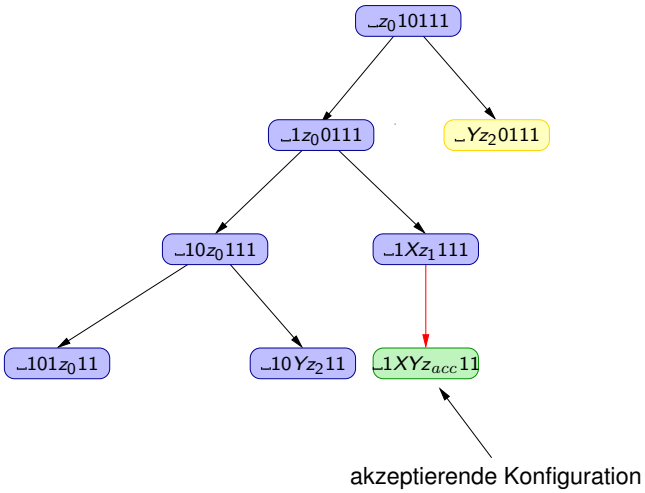


Zurück zum Beispiel (Forts.)

FIFO–Queue:



Ausgewerteter Teil des Berechnungsbaums:



Bemerkungen zum obigen Satz

- Der obige Algorithmus kann mit einer deterministischen Dreiband-Turingmaschine realisiert werden:
  - Band 1 speichert die Eingabe.
  - Band 2 speichert die verkettete Liste.
  - Band 3 dient zur Berechnung der Folgekonfiguration.
- Falls die nichtdeterministische Turingmaschine die Eingabe x akzeptiert, dann findet der Algorithmus einen akzeptierenden Berechnungspfad minimaler Länge.
- Die eingesetzte Technik bezeichnet man als Breitensuche.
- Der Aufwand ist exponentiell zur Laufzeit der nichtdeterministischen Turingmaschine.

Turingmaschinen mit Ausgabe

Aufgabe: Einsatz einer Turingmaschine zur Berechnung einer Funktion $f : \Sigma^* \mapsto \Sigma^*$

Ansatz:

- Die Ausgabe wird auf das Eingabeband geschrieben.
- Zusätzliche Bänder sind erlaubt.
- Die Berechnung ist beendet, wenn sich die Turingmaschine im akzeptierenden oder verwerfenden Zustand befindet.
- Das Ergebnis der Berechnung ist abhängig vom Endzustand:
  - Endzustand z_{acc} $\rightsquigarrow$ Ergebnis ist das Wort auf dem. Eingabeband
  - Endzustand z_{rej} $\rightsquigarrow$ Ergebnis ist undefiniert.

Beispiel

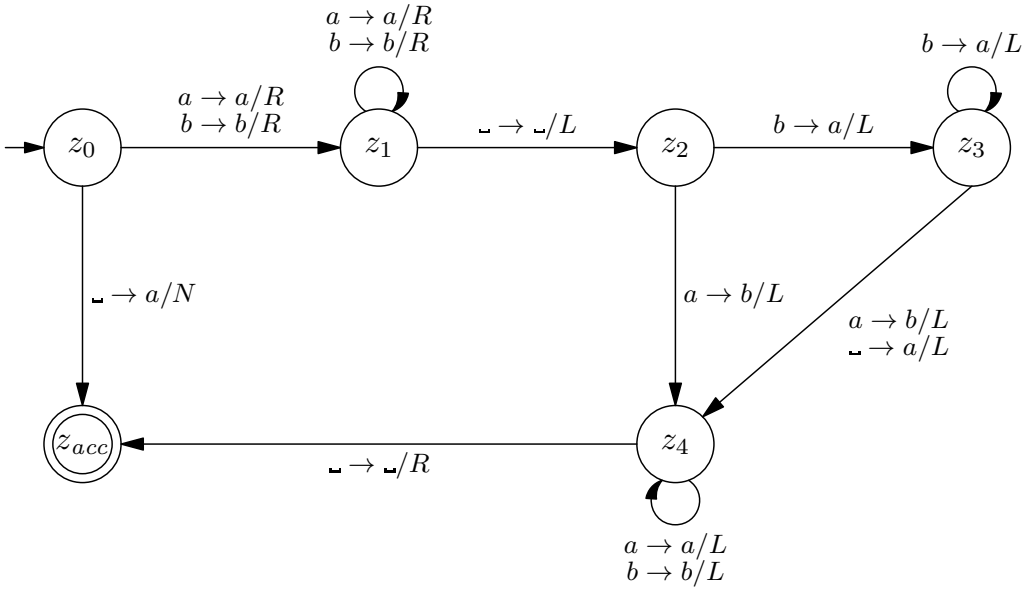
Aufgabe: Berechne für die Eingabe $x \in \{a, b\}^*$ das lexikografisch nächste Wort.

Ansatz: Addition im Binärsystem

Zuordnung:

- $a \rightsquigarrow 0$
- $b \rightsquigarrow 1$

Beispiel (Forts.)



Zusammenfassung

- Die Turingmaschine ist ein einfaches Berechnungsmodell.
- Es gibt verschiedene Arten von Turingmaschinen:
  - Einband $\leftrightarrow$ Mehrband
  - Determinismus $\leftrightarrow$ Nichtdeterminismus
- Lässt man den Ressourcenverbrauch außer Acht, dann sind alle Varianten von Turingmaschinen äquivalent.
- Die Church-Turing These besagt, dass man mit Turingmaschinen jedes algorithmisch lösbare Problem berechnen kann.
- Es gibt weitere Berechnungsmodelle, die zu Turingmaschinen äquivalent sind. Ein Beispiel ist das Modell der Random Access Machine.