Berechenbarkeit und Komplexität

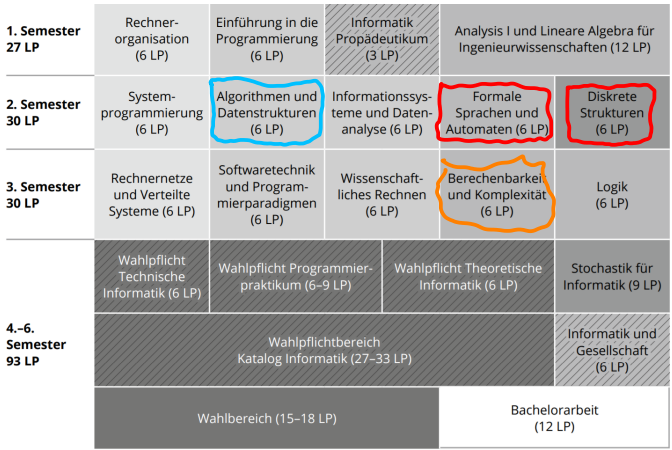
Dozent: Mathias Weller (Skript adaptiert von Rolf Niedermeier)
Betreuer: Leon Kellerhals, Vincent Froese und Philipp Zschoche
Sekretariat: Christlinde Thielcke

Viele Fleißige Tutorinnen und Tutoren

Fachmentorin: Niloofar Nazemi

TU Berlin
Fakultät IV
Fachgebiet Algorithmik und Komplexitätstheorie
https://www.akt.tu-berlin.de

Informatikstudium an der TU Berlin



Organisation

Vorlesungsbetrieb:

- Vorlesung: Screencasts und PDF-Folien verfügbar über ISIS
- “Freie Großübung” + “Modulkonferenz”

Organisation

Vorlesungsbetrieb:

- Vorlesung: Screencasts und PDF-Folien verfügbar über ISIS
- “Freie Großübung” + “Modulkonferenz”

Tutorien:

- Tutorien: siehe ISIS und MOSES
- Tutor*innensprechstunde: TBA

Organisation

Vorlesungsbetrieb:

- Vorlesung: Screencasts und PDF-Folien verfügbar über ISIS
- "Freie Großübung" + "Modulkonferenz"

Tutorien:

- Tutorien: siehe ISIS und MOSES
- Tutor*innensprechstunde: TBA

Prüfungen: Portfolioprüfung

- Multiple-Choice-Test: 25 PP (ca. Mitte Dezember)
- Hausaufgabe in Dreiergruppen 25 PP (im Januar)
- Schriftlicher Test: 50 PP (Termin wird über ISIS bekanntgegben)

Ergänzendes Material

Literatur:

- Uwe Schöning. *Theoretische Informatik–kurz gefasst*. Spektrum Akademischer Verlag 2008 (5. Auflage).
- Elaine Rich. *Automata, Computability, and Complexity*. Pearson 2008.
- Cristopher Moore, Stephan Mertens. *The Nature of Computation*. Oxford University Press 2011.

Weiteres Material:
YouTube-Kanal NLogSpace (https://www.youtube.com/channel/UCMWYg3eBFp5bbqjlllUku_w)

Gliederung

1. Einführung
2. Berechenbarkeitsbegriff
3. LOOP-, WHILE-, und GOTO-Berechenbarkeit
4. Primitive und partielle Rekursion
5. Die Ackermannfunktion
6. (Un-)Entscheidbarkeit, Halteproblem und Reduzierbarkeit
7. Das Postsche Korrespondenzproblem
8. Komplexität – Einführung
9. NP-Vollständigkeit
10. PSPACE

es gibt unentscheidbare Probleme!

was ist praktisch berechenbar

Effizienz!

Das „hello, world“-Problem

Ziel: Entwicklung von Programm E mit folgender Spezifikation:

Input: Programm P
Output: „Top“, wenn P den string „hello, world“ ausgibt, „Flop“, sonst

Das „hello, world“-Problem

Ziel: Entwicklung von Programm E mit folgender Spezifikation:

Input: Programm P ⇝ z.B.: Compiler / Interpreter

Output: „Top“, wenn P den string „hello, world“ ausgibt, „Flop“, sonst

Bemerkung: E hat „Typ höherer Ordnung“ (d.h. Eingabe ist (Text eines) Programms P).

Das „hello, world“-Problem

Ziel: Entwicklung von Programm E mit folgender Spezifikation:

Input: Programm P
Output: „Top“, wenn P den string „hello, world“ ausgibt, „Flop“, sonst
Bemerkung: E hat „Typ höherer Ordnung“ (d.h. Eingabe ist (Text eines) Programms P).

Beispiel für Eingabe P

```
main(){
  printf("hello, world");
}
```

↝ Existiert ein Programm E für diese spezielle Eingabe P? ↝ Top

↝ Existiert ein Programm E auch für **beliebige** Programme P? ↝ nicht möglich ↯

Wiederholung: Endliche Automaten

Definition (Endlicher Automat)

- Ein **(deterministischer) endlicher Automat** (kurz **DFA**) ist ein Quintupel $M = (Z, \Sigma, \delta, z_0, E)$ mit
  - Z ist eine nichtleere, endliche Menge von **Zuständen**,
  - Σ ist ein nichtleeres, endliches Alphabet von **Eingabezeichen** mit $Z \cap \Sigma = \emptyset$,

Wiederholung: Endliche Automaten

Definition (Endlicher Automat)

- Ein **(deterministischer) endlicher Automat** (kurz **DFA**) ist ein Quintupel $M = (Z, \Sigma, \delta, z_0, E)$ mit
  - Z ist eine nichtleere, endliche Menge von **Zuständen**,
  - Σ ist ein nichtleeres, endliches Alphabet von **Eingabezeichen** mit $Z \cap \Sigma = \emptyset$,
  - $\delta : Z \times \Sigma \to Z$ ist die **partielle Überführungsfunktion**,

Wiederholung: Endliche Automaten

Definition (Endlicher Automat)

- Ein **(deterministischer) endlicher Automat** (kurz **DFA**) ist ein Quintupel $M = (Z, \Sigma, \delta, z_0, E)$ mit
  - Z ist eine nichtleere, endliche Menge von **Zuständen**,
  - Σ ist ein nichtleeres, endliches Alphabet von **Eingabezeichen** mit $Z \cap \Sigma = \emptyset$,
  - $\delta\colon Z \times \Sigma \to Z$ ist die **partielle Überführungsfunktion**,
  - $z_0 \in Z$ ist der **Startzustand** und
  - $E \subseteq Z$ ist die Menge der **Endzustände**.

Wiederholung: Endliche Automaten

Definition (Endlicher Automat)

- Ein **(deterministischer) endlicher Automat** (kurz **DFA**) ist ein Quintupel $M = (Z, \Sigma, \delta, z_0, E)$ mit
  - Z ist eine nichtleere, endliche Menge von **Zuständen**,
  - Σ ist ein nichtleeres, endliches Alphabet von **Eingabezeichen** mit $Z \cap \Sigma = \emptyset$,
  - $\delta\colon Z \times \Sigma \to Z$ ist die **partielle Überführungsfunktion**,
  - $z_0 \in Z$ ist der **Startzustand** und
  - $E \subseteq Z$ ist die Menge der **Endzustände**.
- Zu M definieren wir die partielle Funktion $\hat{\delta}\colon Z \times \Sigma^* \to Z$ induktiv für alle $z \in Z$:

$$\hat{\delta}(z, \epsilon) := z$$

$$\forall_{x \in \Sigma^*} \qquad \hat{\delta}(z, ax) := \hat{\delta}(\delta(z, a), x) \qquad \text{falls } \delta(z, a) \neq \perp$$

Wiederholung: Endliche Automaten

Definition (Endlicher Automat)

- Ein **(deterministischer) endlicher Automat** (kurz **DFA**) ist ein Quintupel $M = (Z, \Sigma, \delta, z_0, E)$ mit
  - Z ist eine nichtleere, endliche Menge von **Zuständen**,
  - Σ ist ein nichtleeres, endliches Alphabet von **Eingabezeichen** mit $Z \cap \Sigma = \emptyset$,
  - $\delta\colon Z \times \Sigma \to Z$ ist die **partielle Überführungsfunktion**,
  - $z_0 \in Z$ ist der **Startzustand** und
  - $E \subseteq Z$ ist die Menge der **Endzustände**.
- Zu M definieren wir die partielle Funktion $\hat{\delta}\colon Z \times \Sigma^* \to Z$ induktiv für alle $z \in Z$:

$$\hat{\delta}(z, \epsilon) := z$$

$$\forall_{x \in \Sigma^*} \qquad \hat{\delta}(z, ax) := \hat{\delta}(\delta(z, a), x) \qquad \text{falls } \delta(z, a) \neq \perp$$

- Ein DFA $M = (Z, \Sigma, \delta, z_0, E)$ **akzeptiert** ein Wort $w \in \Sigma^*$ falls $\hat{\delta}(z_0, w) \in E$

Wiederholung: Endliche Automaten

Definition (Endlicher Automat)

- Ein **(deterministischer) endlicher Automat** (kurz **DFA**) ist ein Quintupel $M = (Z, \Sigma, \delta, z_0, E)$ mit
  - Z ist eine nichtleere, endliche Menge von **Zuständen**,
  - Σ ist ein nichtleeres, endliches Alphabet von **Eingabezeichen** mit $Z \cap \Sigma = \emptyset$,
  - $\delta\colon Z \times \Sigma \to Z$ ist die **partielle Überführungsfunktion**,
  - $z_0 \in Z$ ist der **Startzustand** und
  - $E \subseteq Z$ ist die Menge der **Endzustände**.
- Zu M definieren wir die partielle Funktion $\hat{\delta}\colon Z \times \Sigma^* \to Z$ induktiv für alle $z \in Z$:

$$\hat{\delta}(z, \epsilon) := z$$

$$\forall_{x \in \Sigma^*} \qquad \hat{\delta}(z, ax) := \hat{\delta}(\delta(z, a), x) \qquad \text{falls } \delta(z, a) \neq \perp$$

- Ein DFA $M = (Z, \Sigma, \delta, z_0, E)$ **akzeptiert** ein Wort $w \in \Sigma^*$ falls $\hat{\delta}(z_0, w) \in E$
  - Die von M **akzeptierte Sprache** ist $T(M) := \{x \in \Sigma^* \mid \hat{\delta}(z_0, x) \in E\}$.

Wiederholung: Endliche Automaten II

Beispielautomat

$M = (\{z_0, z_1, z_2\}, \{0, 1\}, \delta, z_0, \{z_2\})$ mit

δ	z_0	z_1	z_2
0	z_0	z_2	z_1
→ 1	z_1	z_0	z_2

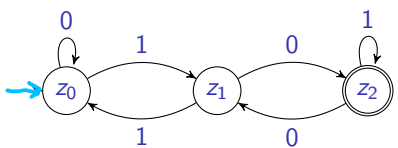
Wiederholung: Endliche Automaten II

Beispielautomat

$M = (\{z_0, z_1, z_2\}, \{0, 1\}, \delta, z_0, \{z_2\})$ mit

δ	z_0	z_1	z_2
0	z_0	z_2	z_1
1	z_1	z_0	z_2

Zustandsgraph



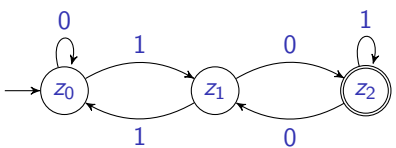
Wiederholung: Endliche Automaten II

Beispielautomat

$M = (\{z_0, z_1, z_2\}, \{0,1\}, \delta, z_0, \{z_2\})$ mit

δ	z_0	z_1	z_2
0	z_0	z_2	z_1
1	z_1	z_0	z_2

Zustandsgraph



$T(M) = \{w \in \{0,1\}^* \mid w \text{ ist Binärdarstellung einer Zahl } n \text{ mit } n \bmod 3 = 2\}$.

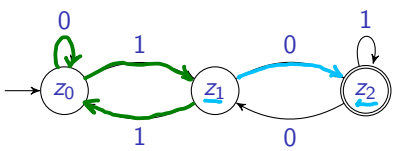
Wiederholung: Endliche Automaten II

Beispielautomat

$M = (\{z_0, z_1, z_2\}, \{0,1\}, \delta, z_0, \{z_2\})$ mit

δ	z_0	z_1	z_2
0	z_0	z_2	z_1
1	z_1	z_0	z_2

Zustandsgraph



$z_i \rightsquigarrow$ das bisher gelesene Wort ist die Binärkodierung einer Zahl n mit Rest i modulo 3.

1

1 1 0 = 6

$n = 3x+1$

$n0 = 2(3x+1)+0 = 6x+2$

$T(M) = \{w \in \{0,1\}^* \mid w \text{ ist Binärdarstellung einer Zahl } n \text{ mit } n \bmod 3 = 2\}$.

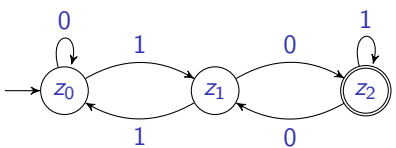
Wiederholung: Endliche Automaten II

Beispielautomat

$M = (\{z_0, z_1, z_2\}, \{0,1\}, \delta, z_0, \{z_2\})$ mit

δ	z_0	z_1	z_2
0	z_0	z_2	z_1
1	z_1	z_0	z_2

Zustandsgraph



$z_i \rightsquigarrow$ das bisher gelesene Wort ist die Binärkodierung einer Zahl n mit Rest i modulo 3.

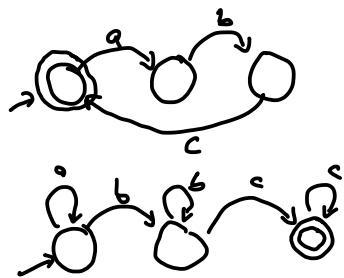
$T(M) = \{w \in \{0,1\}^* \mid w \text{ ist Binärdarstellung einer Zahl } n \text{ mit } n \bmod 3 = 2\}$.

Frage: Sind die Binärdarstellungen der Zahlen n mit $n \bmod 4 = 1$ von einem DFA erkennbar?

Grenzen endlicher Automaten

Gibt es jeweils einen endlichen Automaten zur Erkennung folgender Sprachen?

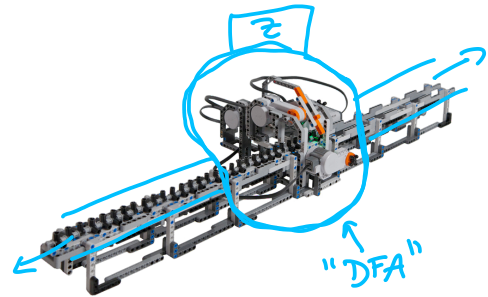
- $\{w \in \{0,1\}^* \mid w \text{ ist Binärdarstellung einer geraden Zahl}\}$ ✓
- $\{a^n b^n \mid 0 \leq n \leq 1000\}$ ✓
- $\{a^n b^n \mid n \geq 0\}$ ✗
- $\{(abc)^n \mid n \geq 0\}$ ✓
- $\{a^n b^m c^k \mid n, m, k \geq 1\}$ ✓
- $\{a^n b^n c^n \mid n \geq 0\}$ ✗
- $\{a^i b^j c^i d^j \mid i, j \geq 0\}$ ✗



Die Turing Maschine

Alan Mathison Turing, 1912-1954.

Inspiration: „Menschliche Computer“ 1890.



LEGO Turing Maschine

Informell

endliche Kontrolle + unendliches Band

Quellen:
http://therunnereclectic.files.wordpress.com/2014/11/alan-turing-running.jpg
http://en.wikipedia.org/wiki/Harvard_Computers
http://cs.cmu.edu/~soonhok/images/20120718_LegoTM/legotm.png

Definition Turing-Maschinen

Definition ((deterministische) Turing-Machine)

Eine **Turing-Maschine (kurz DTM)** ist ein Septupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit

Definition Turing-Maschinen

Definition ((deterministische) Turing-Machine)

Eine **Turing-Maschine (kurz DTM)** ist ein Septupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit

- Z, einer nicht-leeren, endlichen Menge von **Zuständen**,
- Σ, dem **Eingabealphabet**,
- Γ, dem **Bandalphabet**, wobei $\Sigma \subseteq \Gamma$ und $\square \in \Gamma \setminus \Sigma$,
- δ die **partielle Überführungsfunktion**
  $\delta : (Z \setminus E) \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$,
- $z_0 \in Z$, dem **Startzustand**,
- $\square \in \Gamma \setminus \Sigma$, dem **Blanksymbol**,
- $E \subseteq Z$, der Menge der **Endzustände**.

Definition Turing-Maschinen

Definition ((deterministische) Turing-Machine)

Eine **Turing-Maschine (kurz DTM)** ist ein Septupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit

- Z, einer nicht-leeren, endlichen Menge von **Zuständen**,
- Σ, dem **Eingabealphabet**,
- $\Gamma \supseteq \Sigma$, dem **Arbeits-** oder **Bandalphabet** mit $\Gamma \cap Z = \emptyset$,
- δ die **partielle Überführungsfunktion**
- $z_0 \in Z$, dem **Startzustand**,
- $\square \in \Gamma \setminus \Sigma$, dem **Blanksymbol** und
- $E \subseteq Z$, der Menge der **Endzustände**.

Definition Turing-Maschinen

Definition ((deterministische) Turing-Machine)

Eine **Turing-Maschine (kurz DTM)** ist ein Septupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit

- Z, einer nicht-leeren, endlichen Menge von **Zuständen**,
- Σ, dem **Eingabealphabet**,
- $\Gamma \supseteq \Sigma$, dem **Arbeits-** oder **Bandalphabet** mit $\Gamma \cap Z = \emptyset$,
- $\delta\colon (Z \setminus E) \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$, die **partielle Überführungsfunktion**
- $z_0 \in Z$, dem **Startzustand**,
- $\square \in \Gamma \setminus \Sigma$, dem **Blanksymbol** und
- $E \subseteq Z$, der Menge der **Endzustände**.

Definition Turing-Maschinen

Definition ((deterministische) Turing-Machine)

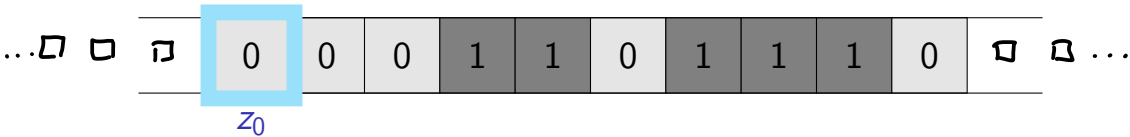
Eine **Turing-Maschine (kurz DTM)** ist ein Septupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit

- Z, einer nicht-leeren, endlichen Menge von **Zuständen**,
- Σ, dem **Eingabealphabet**,
- $\Gamma \supseteq \Sigma$, dem **Arbeits-** oder **Bandalphabet** mit $\Gamma \cap Z = \emptyset$,
- $\delta\colon (Z \setminus E) \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$, die **partielle Überführungsfunktion**
- $z_0 \in Z$, dem **Startzustand**,
- $\square \in \Gamma \setminus \Sigma$, dem **Blanksymbol** und
- $E \subseteq Z$, der Menge der **Endzustände**.

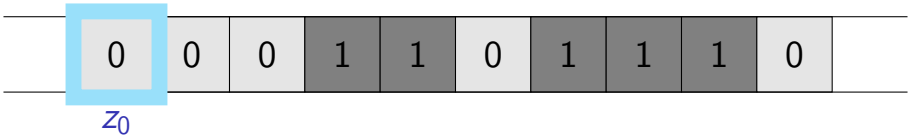
Interpretation: Wenn M im Zustand z das Zeichen a liest und $\delta(z, a) = (z', a', p)$, so

- geht M in Zustand z' über,
- **überschreibt** das a durch ein a'
- bewegt den Lese/Schreibkopf gemäß p (nach **L**inks, **R**echts, oder gar **N**icht)

Arbeitsweise einer Turing-Maschine

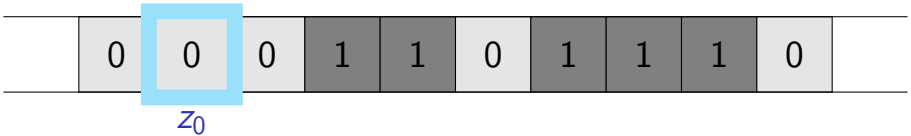


Arbeitsweise einer Turing-Maschine



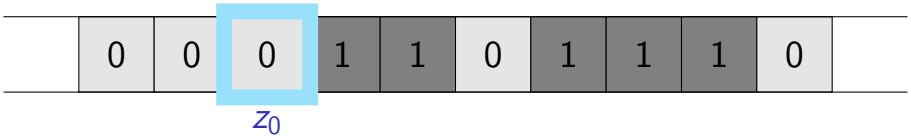
$$\delta(z_0, 0) = (z_0, 0, R)$$

Arbeitsweise einer Turing-Maschine



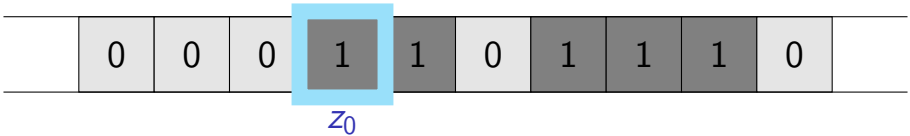
$$\delta(z_0, 0) = (z_0, 0, R)$$

Arbeitsweise einer Turing-Maschine



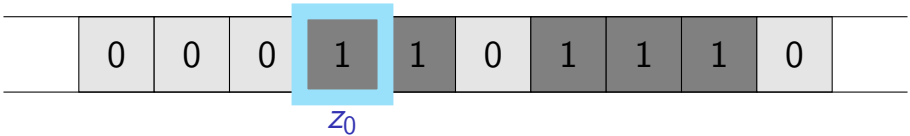
$$\delta(z_0, 0) = (z_0, 0, R)$$

Arbeitsweise einer Turing-Maschine



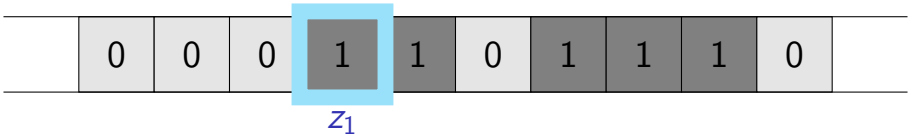
$$\delta(z_0, 0) = (z_0, 0, R)$$

Arbeitsweise einer Turing-Maschine



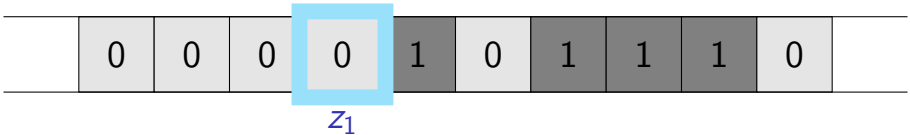
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$

Arbeitsweise einer Turing-Maschine



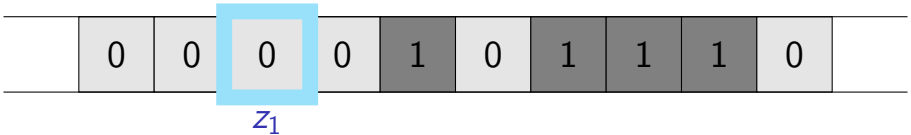
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$

Arbeitsweise einer Turing-Maschine



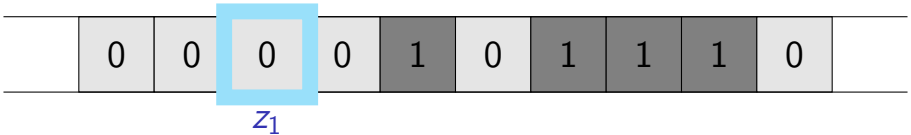
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$

Arbeitsweise einer Turing-Maschine



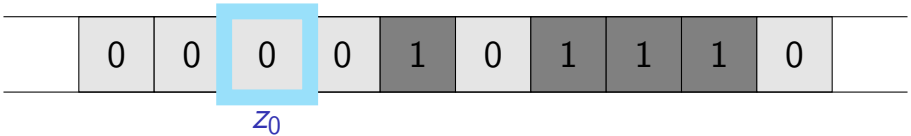
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$

Arbeitsweise einer Turing-Maschine



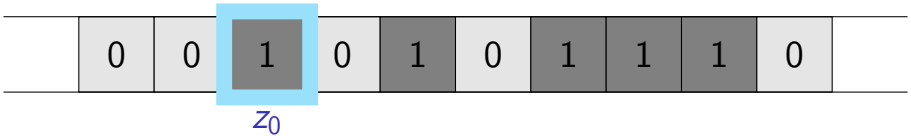
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Arbeitsweise einer Turing-Maschine



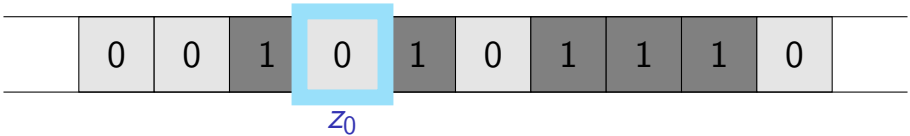
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Arbeitsweise einer Turing-Maschine



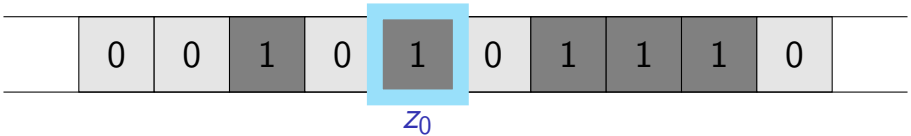
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Arbeitsweise einer Turing-Maschine



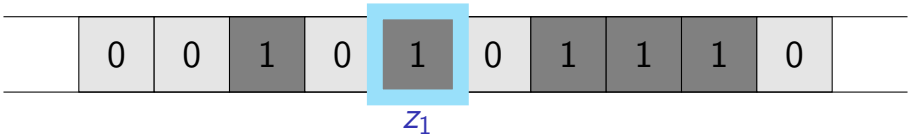
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Arbeitsweise einer Turing-Maschine



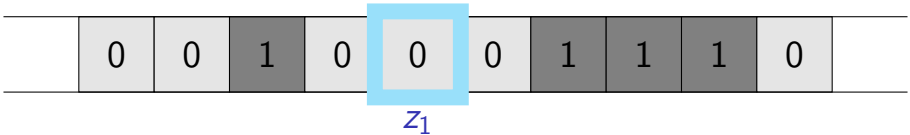
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Arbeitsweise einer Turing-Maschine



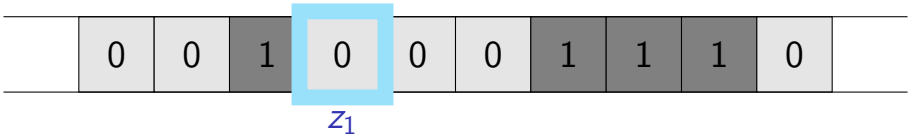
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Arbeitsweise einer Turing-Maschine



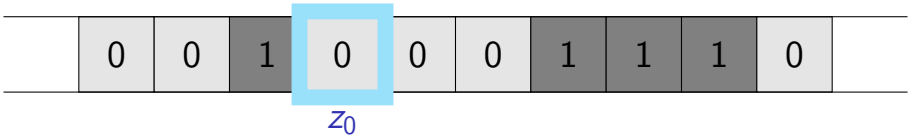
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Arbeitsweise einer Turing-Maschine



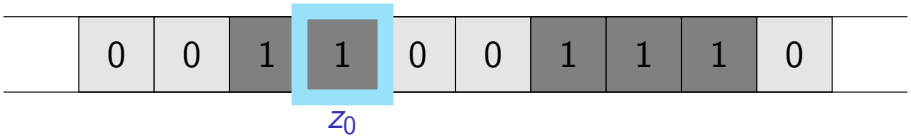
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Arbeitsweise einer Turing-Maschine



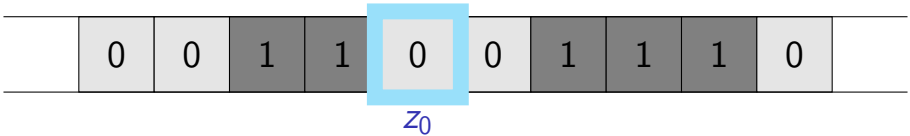
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Arbeitsweise einer Turing-Maschine



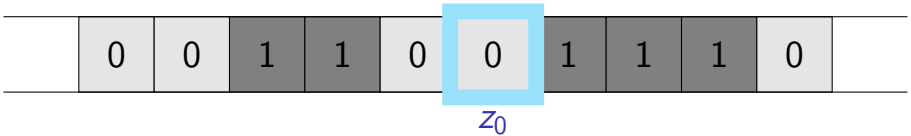
$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Arbeitsweise einer Turing-Maschine



$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Arbeitsweise einer Turing-Maschine



$$\delta(z_0, 0) = (z_0, 0, R)$$
$$\delta(z_0, 1) = (z_1, 0, L)$$
$$\delta(z_1, 0) = (z_0, 1, R)$$

Konfigurationen

Definition (Konfiguration, Folgekonfiguration)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine TM. Eine **Konfiguration** von M ist ein Wort azb mit $a, b \in \Gamma^*$ und $z \in Z$.

Konfigurationen

Definition (Konfiguration, Folgekonfiguration)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine TM. Eine **Konfiguration** von M ist ein Wort azb mit $a, b \in \Gamma^*$ und $z \in Z$.

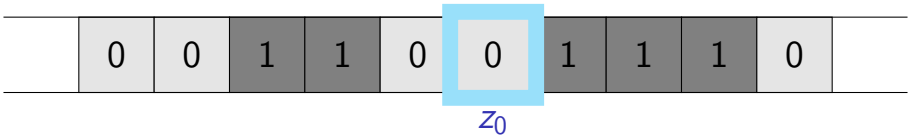
(überflüssige $\square$-Symbole an den Rändern der Konfiguration weglassen)

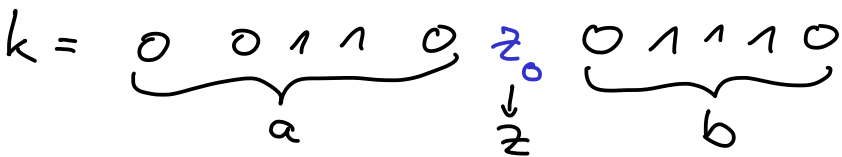
Konfigurationen

Definition (Konfiguration, Folgekonfiguration)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine TM. Eine **Konfiguration** von M ist ein Wort azb mit $a, b \in \Gamma^*$ und $z \in Z$.

(überflüssige $\square$-Symbole an den Rändern der Konfiguration weglassen)





Konfigurationen

Definition (Konfiguration, Folgekonfiguration)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine TM. Eine **Konfiguration** von M ist ein Wort azb mit $a, b \in \Gamma^*$ und $z \in Z$. (überflüssige $\square$-Symbole an den Rändern der Konfiguration weglassen)

Die **Startkonfiguration** zu einem Wort $x \in \Sigma^*$ ist $z_0 x$.

Konfigurationen

Definition (Konfiguration, Folgekonfiguration)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine TM. Eine **Konfiguration** von M ist ein Wort azb mit $a, b \in \Gamma^*$ und $z \in Z$. (überflüssige $\square$-Symbole an den Rändern der Konfiguration weglassen)

Die **Startkonfiguration** zu einem Wort $x \in \Sigma^*$ ist $z_0 x$.

Sei $k = a_1 \dots a_m z b_1 \dots b_n$ eine Konfiguration (falls $n = 0$, dann $b_1 := \square$). Dann

$k \vdash_M^0 k$		
$k \vdash_M^1$	$a_1 \dots a_m z' c b_2 \dots b_n$	falls $\delta(z, b_1) = (z', c, N)$
$k \vdash_M^1$	$a_1 \dots a_m c z' b_2 \dots b_n$	falls $\delta(z, b_1) = (z', c, R)$
$k \vdash_M^1$	$a_1 \dots a_{m-1} z' a_m c b_2 \dots b_n$	falls $\delta(z, b_1) = (z', c, L)$ und $m > 0$
$k \vdash_M^1$	$z' \square c b_2 \dots b_n$	falls $\delta(z, b_1) = (z', c, L)$ und $m = 0$.

Konfigurationen

Definition (Konfiguration, Folgekonfiguration)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine TM. Eine **Konfiguration** von M ist ein Wort azb mit $a, b \in \Gamma^*$ und $z \in Z$. (überflüssige $\square$-Symbole an den Rändern der Konfiguration weglassen)

Die **Startkonfiguration** zu einem Wort $x \in \Sigma^*$ ist $z_0 x$.

Sei $k = a_1 \dots a_m z b_1 \dots b_n$ eine Konfiguration (falls $n = 0$, dann $b_1 := \square$). Dann

$k \vdash_M^0 k$

$k \vdash_M^1 a_1 \dots a_m z' c b_2 \dots b_n$ falls $\delta(z, b_1) = (z', c, N)$

$k \vdash_M^1 a_1 \dots a_m c z' b_2 \dots b_n$ falls $\delta(z, b_1) = (z', c, R)$

$k \vdash_M^1 a_1 \dots a_{m-1} z' a_m c b_2 \dots b_n$ falls $\delta(z, b_1) = (z', c, L)$ und $m > 0$

$k \vdash_M^1 z' \square c b_2 \dots b_n$ falls $\delta(z, b_1) = (z', c, L)$ und $m = 0$.

k ist **haltend** (d.h. k hat keine Folgekonfiguration) falls $\delta(z, b_1) = \perp$

k ist **akzeptierend** falls $z \in E$

Weiter sei $k \vdash_M^{i+1} k' \iff \exists_q k \vdash_M^1 q \vdash_M^i k'$ für alle i und $k \vdash_M^* k' \iff \exists_{i \in \mathbb{N}} k \vdash_M^i k'$

Konfigurationen

Definition (Konfiguration, Folgekonfiguration)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine TM. Eine **Konfiguration** von M ist ein Wort azb mit $a, b \in \Gamma^*$ und $z \in Z$. (überflüssige $\square$-Symbole an den Rändern der Konfiguration weglassen)

Die **Startkonfiguration** zu einem Wort $x \in \Sigma^*$ ist $z_0 x$.

Sei $k = a_1 \dots a_m z b_1 \dots b_n$ eine Konfiguration (falls $n = 0$, dann $b_1 := \square$). Dann

$$k \vdash_M^0 k$$

$$k \vdash_M^1 a_1 \dots a_m z' c b_2 \dots b_n \quad \text{falls } \delta(z, b_1) = (z', c, N)$$

$$k \vdash_M^1 a_1 \dots a_m c z' b_2 \dots b_n \quad \text{falls } \delta(z, b_1) = (z', c, R)$$

$$k \vdash_M^1 a_1 \dots a_{m-1} z' a_m c b_2 \dots b_n \quad \text{falls } \delta(z, b_1) = (z', c, L) \text{ und } m > 0$$

$$k \vdash_M^1 z' \square c b_2 \dots b_n \quad \text{falls } \delta(z, b_1) = (z', c, L) \text{ und } m = 0.$$

k ist **haltend** (d.h. k hat keine Folgekonfiguration) falls $\delta(z, b_1) = \perp$

k ist **akzeptierend** falls $z \in E$

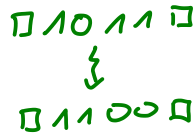
Weiter sei $k \vdash_M^{i+1} k' \iff \exists_q k \vdash_M^1 q \vdash_M^i k'$ für alle i und $k \vdash_M^* k' \iff \exists_{i \in \mathbb{N}} k \vdash_M^i k'$

Frage:gibt es akzeptierende Konfigurationen, die nicht haltend sind?

Beispiel Turing-Maschine: Binärzahl inkrementieren

$M = (\{z_0, z_1, z_2, z_e\}, \Sigma = \{0, 1\}, \Gamma = \{0, 1, \square\}, \delta, z_0, \square, \{z_e\})$

δ	0	1	$\square$
z_0	$(z_0, 0, R)$	$(z_0, 1, R)$	$(z_1, \square, L)$
z_1	$(z_2, 1, L)$	$(z_1, 0, L)$	$(z_e, 1, N)$
z_2	$(z_2, 0, L)$	$(z_2, 1, L)$	$(z_e, \square, R)$

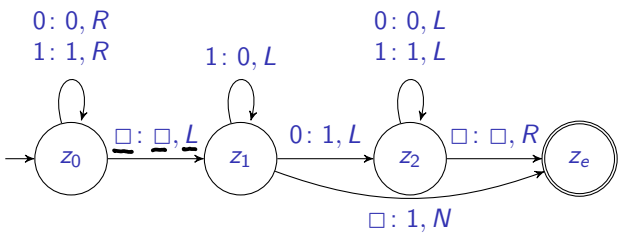


Beispiel Turing-Maschine: Binärzahl inkrementieren

$M = (\{z_0, z_1, z_2, z_e\}, \Sigma = \{0, 1\}, \Gamma = \{0, 1, \square\}, \delta, z_0, \square, \{z_e\})$

δ	0	1	$\square$
z_0	$(z_0, 0, R)$	$(z_0, 1, R)$	$(z_1, \square, L)$
z_1	$(z_2, 1, L)$	$(z_1, 0, L)$	$(z_e, 1, N)$
z_2	$(z_2, 0, L)$	$(z_2, 1, L)$	$(z_e, \square, R)$

Zustandsgraph

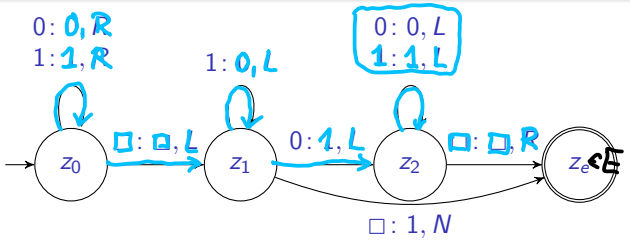


Beispiel Turing-Maschine: Binärzahl inkrementieren

$M = (\{z_0, z_1, z_2, z_e\}, \Sigma = \{0, 1\}, \Gamma = \{0, 1, \square\}, \delta, z_0, \square, \{z_e\})$

δ	0	1	$\square$
z_0	$(z_0, 0, R)$	$(z_0, 1, R)$	$(z_1, \square, L)$
z_1	$(z_2, 1, L)$	$(z_1, 0, L)$	$(z_e, 1, N)$
z_2	$(z_2, 0, L)$	$(z_2, 1, L)$	$(z_e, \square, R)$

Zustandsgraph



Beispiel: Eingabe 101

$z_0 101 \vdash_M^1$

$1 z_0 01 \vdash_M^1$

$10 z_0 1 \vdash_M^1$

$101 z_0 \square \vdash_M^1$

$10 z_1 1 \vdash_M^1$

$1 z_1 00 \vdash_M^1$

$z_2 110 \vdash_M^1$

$z_2 \square 110 \vdash_M^1$

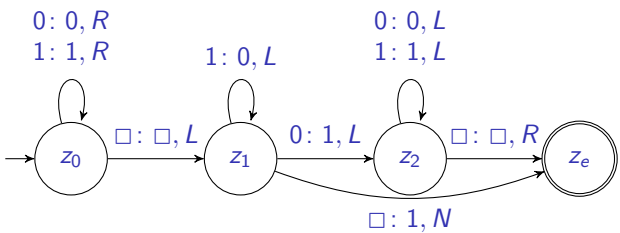
$z_e 110$

Beispiel Turing-Maschine: Binärzahl inkrementieren

$M = (\{z_0, z_1, z_2, z_e\}, \Sigma = \{0,1\}, \Gamma = \{0,1,\square\}, \delta, z_0, \square, \{z_e\})$

δ	0	1	$\square$
z_0	$(z_0, 0, R)$	$(z_0, 1, R)$	$(z_1, \square, L)$
z_1	$(z_2, 1, L)$	$(z_1, 0, L)$	$(z_e, 1, N)$
z_2	$(z_2, 0, L)$	$(z_2, 1, L)$	$(z_e, \square, R)$

Zustandsgraph



Beispiel: Eingabe 101

$$
\begin{aligned}
z_0 101 &\vdash_M^1 \\
1 z_0 01 &\vdash_M^1 \\
10 z_0 1 &\vdash_M^1 \\
101 z_0 &\vdash_M^1 \\
10 z_1 1 &\vdash_M^1 \\
1 z_1 00 &\vdash_M^1 \\
z_2 110 &\vdash_M^1 \\
z_2 \square 110 &\vdash_M^1 \\
z_e 110 &
\end{aligned}
$$

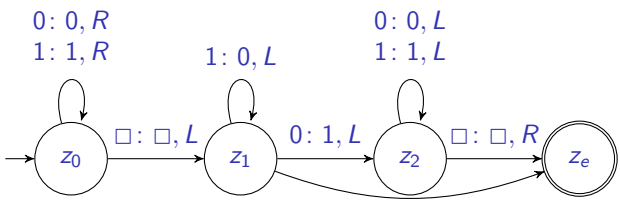
$z_e 110$ haltend & akzeptierend

Beispiel Turing-Maschine: Binärzahl inkrementieren

$M = (\{z_0, z_1, z_2, z_e\}, \Sigma = \{0,1\}, \Gamma = \{0,1,\square\}, \delta, z_0, \square, \{z_e\})$

δ	0	1	$\square$
z_0	$(z_0, 0, R)$	$(z_0, 1, R)$	$(z_1, \square, L)$
z_1	$(z_2, 1, L)$	$(z_1, 0, L)$	$(z_e, 1, N)$
z_2	$(z_2, 0, L)$	$(z_2, 1, L)$	$(z_e, \square, R)$

Zustandsgraph



Frage:Was macht M bei leerer Eingabe? Bei Eingabe 000?

Beispiel: Eingabe 101

$$
\begin{aligned}
z_0101 &\vdash_M^1 \\
1z_001 &\vdash_M^1 \\
10z_01 &\vdash_M^1 \\
101z_0 &\vdash_M^1 \\
10z_11 &\vdash_M^1 \\
1z_100 &\vdash_M^1 \\
z_2110 &\vdash_M^1 \\
z_2\square110 &\vdash_M^1 \\
z_e110 &
\end{aligned}
$$

z_e110 haltend & akzeptierend

Akzeptieren und Halten einer TM

Definition (Akzeptieren, Halten)

Turing-Maschine $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$:

- M **hält** auf $w \in \Sigma^*$, falls eine **haltende** Konfiguration k' existiert mit $z_0 w \vdash_M^* k'$.
- M **akzeptiert** $w \in \Sigma^*$, falls eine **akzeptierende** Konfiguation k' existiert mit $z_0 w \vdash_M^* k'$.
- M **akzeptiert Sprache** $T(M)$ enthält genau die Wörter w die M akzeptiert. Formal,

$$T(M) := \{w \in \Sigma^* \mid \exists_{\alpha,\beta \in \Gamma^*} \exists_{z \in E} : z_0 w \vdash_M^* \alpha z \beta\}.$$

Beispiel Turing-Maschine: akzeptiere $\{0^n1^n \mid n \in \mathbb{N}\}$

$M = (\{z_0, z_1, z_R, z_L, z_e\}, \Sigma = \{0,1\}, \Gamma = \{0,1,\square\}, \delta, z_0, \square, \{z_e\})$

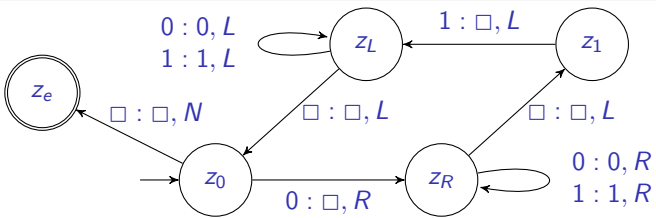
δ	0	1	$\square$
z_0	$(z_R, \square, R)$	$\perp$	$(z_e, \square, N)$
z_1	$\perp$	$(z_L, \square, L)$	$\perp$
z_R	$(z_R, 0, R)$	$(z_R, 1, R)$	$(z_1, \square, L)$
z_L	$(z_L, 0, L)$	$(z_L, 1, L)$	$(z_0, \square, R)$

Beispiel Turing-Maschine: akzeptiere $\{0^n1^n \mid n \in \mathbb{N}\}$

$M = (\{z_0, z_1, z_R, z_L, z_e\}, \Sigma = \{0,1\}, \Gamma = \{0,1,\square\}, \delta, z_0, \square, \{z_e\})$

δ	0	1	$\square$
z_0	$(z_R, \square, R)$	$\bot$	$(z_e, \square, N)$
z_1	$\bot$	$(z_L, \square, L)$	$\bot$
z_R	$(z_R, 0, R)$	$(z_R, 1, R)$	$(z_1, \square, L)$
z_L	$(z_L, 0, L)$	$(z_L, 1, L)$	$(z_0, \square, R)$

Zustandsgraph

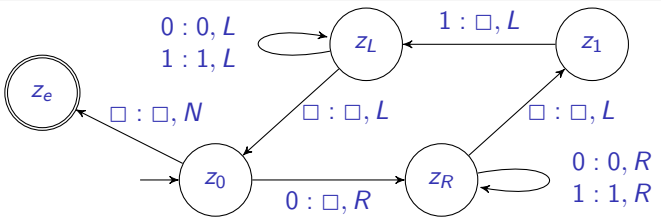


Beispiel Turing-Maschine: akzeptiere $\{0^n1^n \mid n \in \mathbb{N}\}$

$M = (\{z_0, z_1, z_R, z_L, z_e\}, \Sigma = \{0,1\}, \Gamma = \{0,1,\square\}, \delta, z_0, \square, \{z_e\})$

δ	0	1	$\square$
z_0	$(z_R, \square, R)$	$\perp$	$(z_e, \square, N)$
z_1	$\perp$	$(z_L, \square, L)$	$\perp$
z_R	$(z_R, 0, R)$	$(z_R, 1, R)$	$(z_1, \square, L)$
z_L	$(z_L, 0, L)$	$(z_L, 1, L)$	$(z_0, \square, R)$

Zustandsgraph



Beispiel: Eingabe 0011

$$
\begin{aligned}
z_0 0011 &\vdash_M^1 \\
0 z_R 11 &\vdash_M^2 \\
011 z_R &\vdash_M^1 \\
01 z_1 1 &\vdash_M^1 \\
0 z_L 1 &\vdash_M^2 \\
z_L \square 01 &\vdash_M^1 \\
z_0 01 &\vdash_M^1 \\
z_R 1 &\vdash_M^1 \\
1 z_R &\vdash_M^1 \\
z_1 1 &\vdash_M^1 \\
z_L &\vdash_M^1 \\
z_0 &\vdash_M^1 \\
z_e &
\end{aligned}
$$