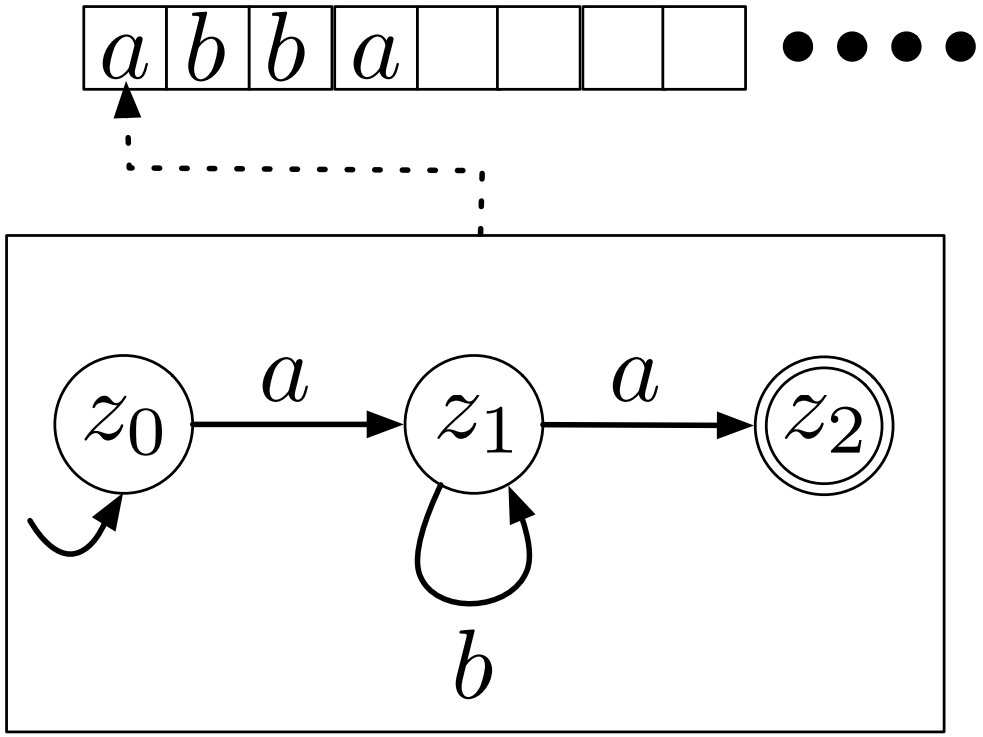
Formale Grundlagen der Informatik 1
Kapitel 7
Turing-Maschinen

Frank Heitmann
heitmann@informatik.uni-hamburg.de

28. April 2014

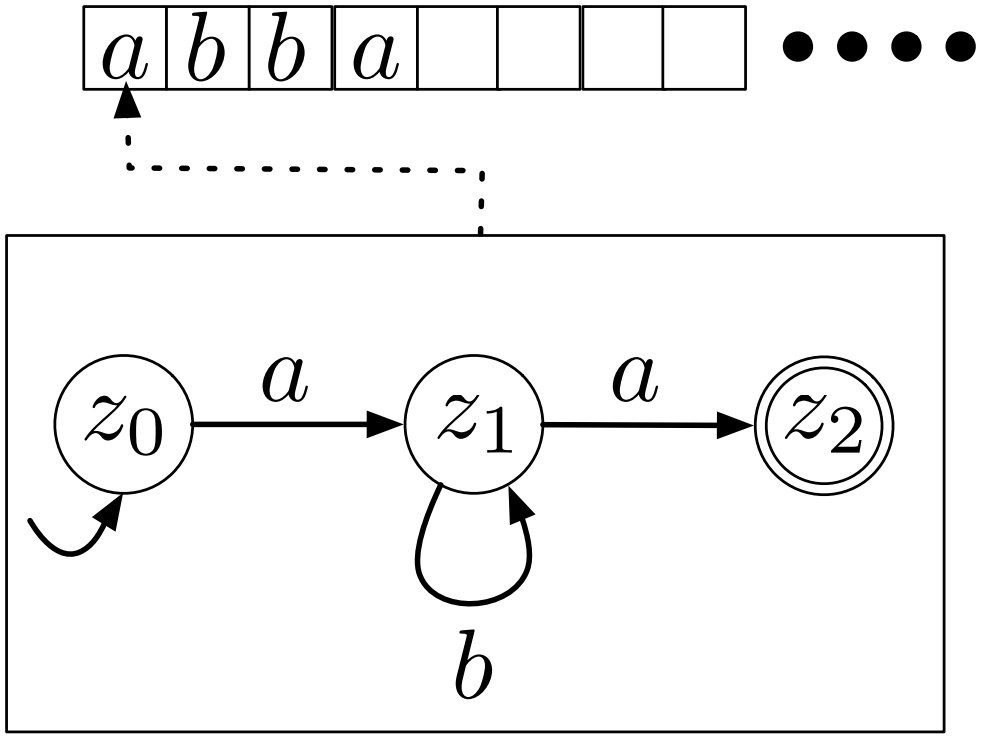
Vom DFA zur TM



Wir wollen

- auf dem Band nach rechts **und** links gehen können und
- auf dem Band lesen **und** schreiben können.

Vom DFA zur TM



Wir wollen

- auf dem Band nach rechts **und** links gehen können und
- auf dem Band lesen **und** schreiben können.

Bedeutung der TM

Das genügt schon, um zur **Turing-Maschine** (TM) zu gelangen.

Die Turing-Maschine gilt als abstraktes Modell dessen, was von jeder Maschine und jedem Menschen berechenbar ist.

Was eine Turing-Maschine nicht kann, kann also auch kein Mensch! (So die **Church-Turing-These**...)

Motivation

Sie ist daher geeignet, um sich über Möglichkeiten und Grenzen des Berechenbaren Gedanken zu machen.

Der Lese-/Schreibkopf

Anmerkung

- Während DFA und PDA nur lesen konnten, kann die TM auch schreiben. Sie hat dazu einen **Lese-/Schreibkopf** (LSK). Dieser tritt (wie der Lesekopf beim DFA) nicht in der formalen Beschreibung auf. In der informalen ist er aber wichtig.
- Der LSK ist stets über dem aktuellem Symbol auf dem Band positioniert.
- Das (Eingabe-)Band ist nach rechts hin unendlich.

Der Lese-/Schreibkopf

Anmerkung

- Während DFA und PDA nur lesen konnten, kann die TM auch schreiben. Sie hat dazu einen **Lese-/Schreibkopf** (LSK). Dieser tritt (wie der Lesekopf beim DFA) nicht in der formalen Beschreibung auf. In der informalen ist er aber wichtig.
- Der LSK ist stets über dem aktuellem Symbol auf dem Band positioniert.
- Das (Eingabe-)Band ist nach rechts hin unendlich.

Der Lese-/Schreibkopf

Anmerkung

- Während DFA und PDA nur lesen konnten, kann die TM auch schreiben. Sie hat dazu einen **Lese-/Schreibkopf** (LSK). Dieser tritt (wie der Lesekopf beim DFA) nicht in der formalen Beschreibung auf. In der informalen ist er aber wichtig.
- Der LSK ist stets über dem aktuellem Symbol auf dem Band positioniert.
- Das (Eingabe-)Band ist nach rechts hin unendlich.

TM formal

Definition (Deterministische TM)

Eine **deterministische Turing-Maschine** (kurz DTM) ist ein 6-Tupel $A = (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$ mit

- Der endlichen Menge von **Zuständen** Z.
- Dem endlichen Alphabet Σ von **Eingabesymbolen**.
- Dem endlichen Alphabet Γ von **Bandsymbolen**, wobei $\Gamma \supsetneq \Sigma$ und $\Gamma \cap Z = \emptyset$ gilt.
- Der (partiellen) **Überführungsfunktion** $\delta : (Z \times \Gamma) \to (\Gamma \times \{L, R, H\} \times Z)$.
- Dem **Startzustand** $z_0 \in Z$.
- Der Menge der **Endzustände** $Z_{end} \subseteq Z$.
- Dem **Symbol für das leere Feld** $\# \in \Gamma \setminus \Sigma$.

TM formal

Definition (Deterministische TM)

Eine **deterministische Turing-Maschine** (kurz DTM) ist ein 6-Tupel $A = (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$ mit

- Der endlichen Menge von **Zuständen** Z.
- Dem endlichen Alphabet Σ von **Eingabesymbolen**.
- Dem endlichen Alphabet Γ von **Bandsymbolen**, wobei $\Gamma \supsetneq \Sigma$ und $\Gamma \cap Z = \emptyset$ gilt.
- Der (partiellen) **Überführungsfunktion** $\delta : (Z \times \Gamma) \to (\Gamma \times \{L, R, H\} \times Z)$.
- Dem **Startzustand** $z_0 \in Z$.
- Der Menge der **Endzustände** $Z_{end} \subseteq Z$.
- Dem **Symbol für das leere Feld** $\# \in \Gamma \setminus \Sigma$.

TM formal

Definition (Deterministische TM)

Eine **deterministische Turing-Maschine** (kurz DTM) ist ein 6-Tupel $A = (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$ mit

- Der endlichen Menge von **Zuständen** Z.
- Dem endlichen Alphabet Σ von **Eingabesymbolen**.
- Dem endlichen Alphabet Γ von **Bandsymbolen**, wobei $\Gamma \supsetneq \Sigma$ und $\Gamma \cap Z = \emptyset$ gilt.
- Der (partiellen) **Überführungsfunktion** $\delta : (Z \times \Gamma) \to (\Gamma \times \{L, R, H\} \times Z)$.
- Dem **Startzustand** $z_0 \in Z$.
- Der Menge der **Endzustände** $Z_{end} \subseteq Z$.
- Dem **Symbol für das leere Feld** $\# \in \Gamma \setminus \Sigma$.

TM formal

Definition (Deterministische TM)

Eine **deterministische Turing-Maschine** (kurz DTM) ist ein 6-Tupel $A = (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$ mit

- Der endlichen Menge von **Zuständen** Z.
- Dem endlichen Alphabet Σ von **Eingabesymbolen**.
- Dem endlichen Alphabet Γ von **Bandsymbolen**, wobei $\Gamma \supsetneq \Sigma$ und $\Gamma \cap Z = \emptyset$ gilt.
- Der (partiellen) **Überführungsfunktion** $\delta : (Z \times \Gamma) \to (\Gamma \times \{L, R, H\} \times Z)$.
- Dem **Startzustand** $z_0 \in Z$.
- Der Menge der **Endzustände** $Z_{end} \subseteq Z$.
- Dem **Symbol für das leere Feld** $\# \in \Gamma \setminus \Sigma$.

TM formal

Definition (Deterministische TM)

Eine **deterministische Turing-Maschine** (kurz DTM) ist ein 6-Tupel $A = (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$ mit

- Der endlichen Menge von **Zuständen** Z.
- Dem endlichen Alphabet Σ von **Eingabesymbolen**.
- Dem endlichen Alphabet Γ von **Bandsymbolen**, wobei $\Gamma \supsetneq \Sigma$ und $\Gamma \cap Z = \emptyset$ gilt.
- Der (partiellen) **Überführungsfunktion** $\delta : (Z \times \Gamma) \to (\Gamma \times \{L, R, H\} \times Z)$.
- Dem **Startzustand** $z_0 \in Z$.
- Der Menge der **Endzustände** $Z_{end} \subseteq Z$.
- Dem **Symbol für das leere Feld** $\# \in \Gamma \setminus \Sigma$.

TM formal

Definition (Deterministische TM)

Eine **deterministische Turing-Maschine** (kurz DTM) ist ein 6-Tupel $A = (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$ mit

- Der endlichen Menge von **Zuständen** Z.
- Dem endlichen Alphabet Σ von **Eingabesymbolen**.
- Dem endlichen Alphabet Γ von **Bandsymbolen**, wobei $\Gamma \supsetneq \Sigma$ und $\Gamma \cap Z = \emptyset$ gilt.
- Der (partiellen) **Überführungsfunktion** $\delta : (Z \times \Gamma) \to (\Gamma \times \{L, R, H\} \times Z)$.
- Dem **Startzustand** $z_0 \in Z$.
- Der Menge der **Endzustände** $Z_{end} \subseteq Z$.
- Dem **Symbol für das leere Feld** $\# \in \Gamma \setminus \Sigma$.

TM formal

Definition (Deterministische TM)

Eine **deterministische Turing-Maschine** (kurz DTM) ist ein 6-Tupel $A = (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$ mit

- Der endlichen Menge von **Zuständen** Z.
- Dem endlichen Alphabet Σ von **Eingabesymbolen**.
- Dem endlichen Alphabet Γ von **Bandsymbolen**, wobei $\Gamma \supsetneq \Sigma$ und $\Gamma \cap Z = \emptyset$ gilt.
- Der (partiellen) **Überführungsfunktion** $\delta : (Z \times \Gamma) \to (\Gamma \times \{L, R, H\} \times Z)$.
- Dem **Startzustand** $z_0 \in Z$.
- Der Menge der **Endzustände** $Z_{end} \subseteq Z$.
- Dem **Symbol für das leere Feld** $\# \in \Gamma \setminus \Sigma$.

TM formal (alternative)

Kantenrelation

Statt der Überführungsfunktion δ kann auch mit einer Relation

$$K \subseteq Z \times \Gamma \times \Gamma \times \{L, R, H\} \times Z$$

gearbeitet werden.

Dabei gibt es dann im deterministischem Fall aber

- zu jedem Pärchen $(z, x) \in Z \times \Gamma$
- nur maximal ein Tripel (x', B, z') derart, dass $(z, x, x', B, z') \in K$ gilt.

TM formal (alternative)

Kantenrelation

Statt der Überführungsfunktion δ kann auch mit einer Relation

$$K \subseteq Z \times \Gamma \times \Gamma \times \{L, R, H\} \times Z$$

gearbeitet werden.
Dabei gibt es dann im deterministischem Fall aber

- zu jedem Pärchen $(z, x) \in Z \times \Gamma$
- nur maximal ein Tripel (x', B, z') derart, dass $(z, x, x', B, z') \in K$ gilt.

TM formal (alternative)

Kantenrelation

Statt der Überführungsfunktion δ kann auch mit einer Relation

$$K \subseteq Z \times \Gamma \times \Gamma \times \{L, R, H\} \times Z$$

gearbeitet werden.
Dabei gibt es dann im deterministischem Fall aber

- zu jedem Pärchen $(z, x) \in Z \times \Gamma$
- nur maximal ein Tripel (x', B, z') derart, dass $(z, x, x', B, z') \in K$ gilt.

Die Überführungsfunktion

Mit $\delta(z, x) = (x', B, z')$ ist gemeint, dass, wenn

- die TM im Zustand z ist und
- der LSK gerade über einem Feld mit dem Symbol x ist,
- nun das x durch x' überschrieben wird,
- dann der LSK nach $B \in \{L, R, H\}$ bewegt wird
  - L ist nach links bewegen,
  - R nach rechts und
  - H den LSK an der Stelle halten
- und in den Zustand z' gewechselt wird.

Die Überführungsfunktion

Mit $\delta(z, x) = (x', B, z')$ ist gemeint, dass, wenn

- die TM im Zustand z ist und
- der LSK gerade über einem Feld mit dem Symbol x ist,
- nun das x durch x' überschrieben wird,
- dann der LSK nach $B \in \{L, R, H\}$ bewegt wird
  - L ist nach links bewegen,
  - R nach rechts und
  - H den LSK an der Stelle halten
- und in den Zustand z' gewechselt wird.

Die Überführungsfunktion

Mit $\delta(z, x) = (x', B, z')$ ist gemeint, dass, wenn

- die TM im Zustand z ist und
- der LSK gerade über einem Feld mit dem Symbol x ist,
- nun das x durch x' überschrieben wird,
- dann der LSK nach $B \in \{L, R, H\}$ bewegt wird
  - L ist nach links bewegen,
  - R nach rechts und
  - H den LSK an der Stelle halten
- und in den Zustand z' gewechselt wird.

Die Überführungsfunktion

Mit $\delta(z, x) = (x', B, z')$ ist gemeint, dass, wenn

- die TM im Zustand z ist und
- der LSK gerade über einem Feld mit dem Symbol x ist,
- nun das x durch x' überschrieben wird,
- dann der LSK nach $B \in \{L, R, H\}$ bewegt wird
  - L ist nach links bewegen,
  - R nach rechts und
  - H den LSK an der Stelle halten
- und in den Zustand z' gewechselt wird.

Die Überführungsfunktion

Mit $\delta(z, x) = (x', B, z')$ ist gemeint, dass, wenn

- die TM im Zustand z ist und
- der LSK gerade über einem Feld mit dem Symbol x ist,
- nun das x durch x' überschrieben wird,
- dann der LSK nach $B \in \{L, R, H\}$ bewegt wird
  - L ist nach links bewegen,
  - R nach rechts und
  - H den LSK an der Stelle halten
- und in den Zustand z' gewechselt wird.

TM informal

Zusammengefasst:

- Die TM hat endlich viele Zustände.
- Sie hat ein Eingabealphabet, aus dessen Buchstaben die Eingabewörter aufgebaut sind.
- Sie hat ein (größeres) Bandalphabet, mit weiteren Symbolen, die benutzt werden können. Dieses enthält mindestens noch das spezielle Symbol #.
- Das Eingabeband ist nach rechts hin unendlich. Zu Anfang steht das Eingabewort ganz links auf diesem Band. Danach folgen unendlich viele #.
- Die TM liest nicht nur das Eingabewort, sondern kann auch auf dem Eingabeband mit dem LSK nach links und rechts wandern und die Symbole manipulieren. Anders als der PDA kommt sie dabei an jedes Symbol heran und nicht nur an ein spezielles (das oberste des Kellers beim PDA)!

TM informal

Zusammengefasst:

- Die TM hat endlich viele Zustände.
- Sie hat ein Eingabealphabet, aus dessen Buchstaben die Eingabewörter aufgebaut sind.
- Sie hat ein (größeres) Bandalphabet, mit weiteren Symbolen, die benutzt werden können. Dieses enthält mindestens noch das spezielle Symbol #.
- Das Eingabeband ist nach rechts hin unendlich. Zu Anfang steht das Eingabewort ganz links auf diesem Band. Danach folgen unendlich viele #.
- Die TM liest nicht nur das Eingabewort, sondern kann auch auf dem Eingabeband mit dem LSK nach links und rechts wandern und die Symbole manipulieren. Anders als der PDA kommt sie dabei an jedes Symbol heran und nicht nur an ein spezielles (das oberste des Kellers beim PDA)!

TM informal

Zusammengefasst:

- Die TM hat endlich viele Zustände.
- Sie hat ein Eingabealphabet, aus dessen Buchstaben die Eingabewörter aufgebaut sind.
- Sie hat ein (größeres) Bandalphabet, mit weiteren Symbolen, die benutzt werden können. Dieses enthält mindestens noch das spezielle Symbol #.
- Das Eingabeband ist nach rechts hin unendlich. Zu Anfang steht das Eingabewort ganz links auf diesem Band. Danach folgen unendlich viele #.
- Die TM liest nicht nur das Eingabewort, sondern kann auch auf dem Eingabeband mit dem LSK nach links und rechts wandern und die Symbole manipulieren. Anders als der PDA kommt sie dabei an jedes Symbol heran und nicht nur an ein spezielles (das oberste des Kellers beim PDA)!

TM informal

Zusammengefasst:

- Die TM hat endlich viele Zustände.
- Sie hat ein Eingabealphabet, aus dessen Buchstaben die Eingabewörter aufgebaut sind.
- Sie hat ein (größeres) Bandalphabet, mit weiteren Symbolen, die benutzt werden können. Dieses enthält mindestens noch das spezielle Symbol #.
- Das Eingabeband ist nach rechts hin unendlich. Zu Anfang steht das Eingabewort ganz links auf diesem Band. Danach folgen unendlich viele #.
- Die TM liest nicht nur das Eingabewort, sondern kann auch auf dem Eingabeband mit dem LSK nach links und rechts wandern und die Symbole manipulieren. Anders als der PDA kommt sie dabei an jedes Symbol heran und nicht nur an ein spezielles (das oberste des Kellers beim PDA)!

TM informal

Zusammengefasst:

- Die TM hat endlich viele Zustände.
- Sie hat ein Eingabealphabet, aus dessen Buchstaben die Eingabewörter aufgebaut sind.
- Sie hat ein (größeres) Bandalphabet, mit weiteren Symbolen, die benutzt werden können. Dieses enthält mindestens noch das spezielle Symbol #.
- Das Eingabeband ist nach rechts hin unendlich. Zu Anfang steht das Eingabewort ganz links auf diesem Band. Danach folgen unendlich viele #.
- Die TM liest nicht nur das Eingabewort, sondern kann auch auf dem Eingabeband mit dem LSK nach links und rechts wandern und die Symbole manipulieren. Anders als der PDA kommt sie dabei an jedes Symbol heran und nicht nur an ein spezielles (das oberste des Kellers beim PDA)!

Konfiguration einer TM

Definition (Konfiguration)

Ein Wort $w \in \Gamma^* \cdot Z \cdot \Gamma^*$ heißt **Konfiguration** der TM $A := (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$. Ist $w = uzv$ mit $z \in Z$ und $u, v \in \Gamma^*$, dann ist

- A im Zustand z,
- die Bandinschrift ist uv (links/rechts davon nur $\#$)
- und das erste Symbol von v ist unter dem LSK. (Ist $v = \lambda$, so ist $\#$ unter dem LSK. Ist $v \neq \lambda$, so ist $v \in \Gamma^*(\Gamma \setminus \{\#\})$.)

Die Menge aller Konfigurationen der TM M ist KONF_M.

Konfiguration einer TM

Definition (Konfiguration)

Ein Wort $w \in \Gamma^* \cdot Z \cdot \Gamma^*$ heißt **Konfiguration** der TM $A := (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$. Ist $w = uzv$ mit $z \in Z$ und $u, v \in \Gamma^*$, dann ist

- A im Zustand z,
- die Bandinschrift ist uv (links/rechts davon nur $\#$)
- und das erste Symbol von v ist unter dem LSK. (Ist $v = \lambda$, so ist $\#$ unter dem LSK. Ist $v \neq \lambda$, so ist $v \in \Gamma^*(\Gamma \setminus \{\#\})$.)

Die Menge aller Konfigurationen der TM M ist KONF_M.

Konfiguration einer TM

Definition (Konfiguration)

Ein Wort $w \in \Gamma^* \cdot Z \cdot \Gamma^*$ heißt **Konfiguration** der TM $A := (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$. Ist $w = uzv$ mit $z \in Z$ und $u, v \in \Gamma^*$, dann ist

- A im Zustand z,
- die Bandinschrift ist uv (links/rechts davon nur $\#$)
- und das erste Symbol von v ist unter dem LSK. (Ist $v = \lambda$, so ist $\#$ unter dem LSK. Ist $v \neq \lambda$, so ist $v \in \Gamma^*(\Gamma \setminus \{\#\})$.)

Die Menge aller Konfigurationen der TM M ist KONF_M.

Konfiguration einer TM

Definition (Konfiguration)

Ein Wort $w \in \Gamma^* \cdot Z \cdot \Gamma^*$ heißt **Konfiguration** der TM $A := (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$. Ist $w = uzv$ mit $z \in Z$ und $u, v \in \Gamma^*$, dann ist

- A im Zustand z,
- die Bandinschrift ist uv (links/rechts davon nur $\#$)
- und das erste Symbol von v ist unter dem LSK. (Ist $v = \lambda$, so ist $\#$ unter dem LSK. Ist $v \neq \lambda$, so ist $v \in \Gamma^*(\Gamma \setminus \{\#\})$.)

Die Menge aller Konfigurationen der TM M ist KONF_M.

Konfiguration einer TM

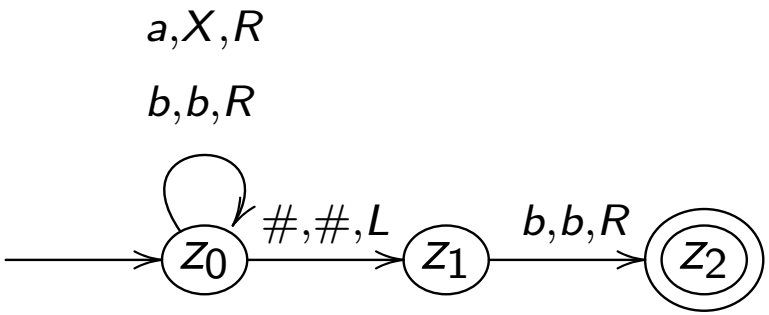
Definition (Konfiguration)

Ein Wort $w \in \Gamma^* \cdot Z \cdot \Gamma^*$ heißt **Konfiguration** der TM $A := (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$. Ist $w = uzv$ mit $z \in Z$ und $u, v \in \Gamma^*$, dann ist

- A im Zustand z,
- die Bandinschrift ist uv (links/rechts davon nur $\#$)
- und das erste Symbol von v ist unter dem LSK. (Ist $v = \lambda$, so ist $\#$ unter dem LSK. Ist $v \neq \lambda$, so ist $v \in \Gamma^*(\Gamma \setminus \{\#\})$.)

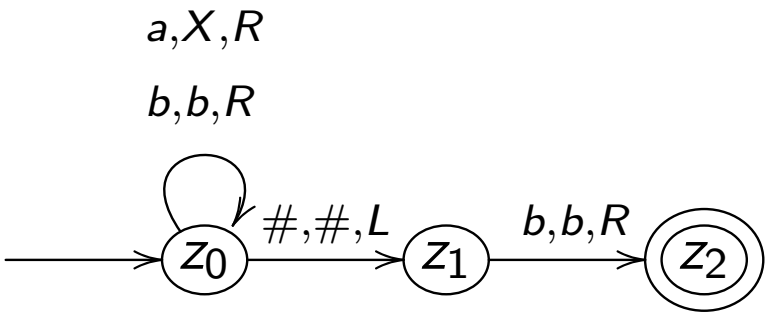
Die Menge aller Konfigurationen der TM M ist KONF_M.

TM: Beispiel



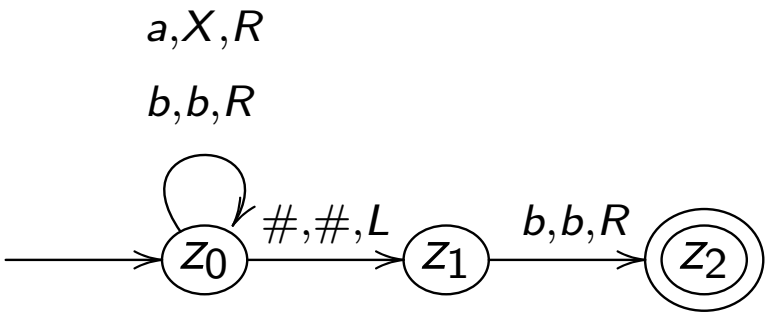
$$z_0aab \vdash Xz_0ab \vdash XXz_0b \vdash XXbz_0\# \vdash XXz_1b \vdash XXbz_2\#$$

TM: Beispiel



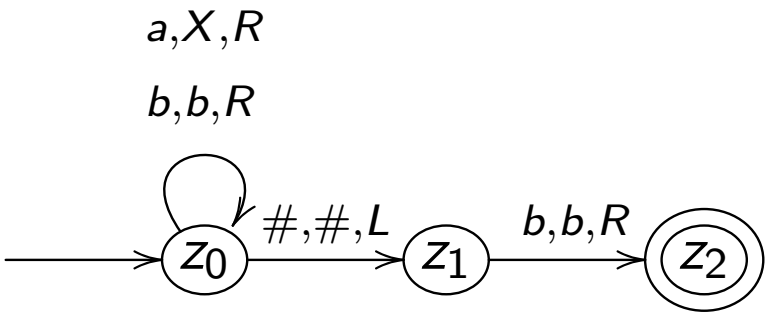
$$z_0aab \vdash Xz_0ab \vdash XXz_0b \vdash XXbz_0\# \vdash XXz_1b \vdash XXbz_2\#$$

TM: Beispiel



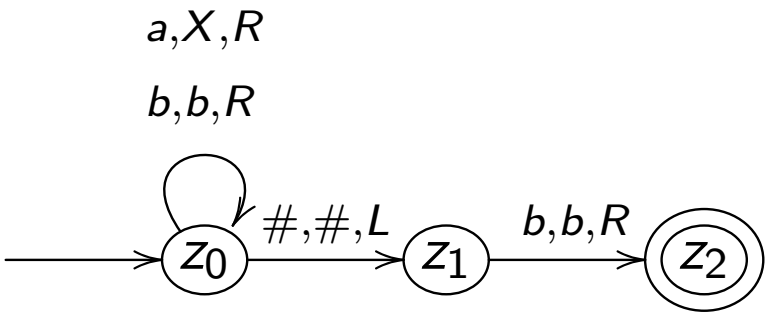
$$z_0aab \vdash Xz_0ab \vdash XXz_0b \vdash XXbz_0\# \vdash XXz_1b \vdash XXbz_2\#$$

TM: Beispiel



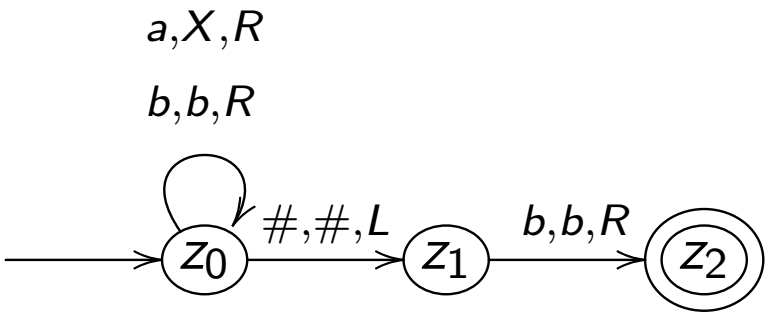
$$z_0aab \vdash Xz_0ab \vdash XXz_0b \vdash XXbz_0\# \vdash XXz_1b \vdash XXbz_2\#$$

TM: Beispiel



$$z_0aab \vdash Xz_0ab \vdash XXz_0b \vdash XXbz_0\# \vdash XXz_1b \vdash XXbz_2\#$$

TM: Beispiel



$$z_0aab \vdash Xz_0ab \vdash XXz_0b \vdash XXbz_0\# \vdash XXz_1b \vdash XXbz_2\#$$

Rechnung einer TM

Die Definition einer Rechnung wird dadurch verkompliziert, dass

- das (Eingabe-)Band einseitig (nach rechts) unendlich ist
- der LSK also nicht über den linken Rand bewegt werden kann
- und die Konfigurationen stets den "relevanten Teil" auf dem Eingabeband darstellen sollen (d.h. insb. das nicht alle # dargestellt werden, sondern nur jene zwischen "richtigen" Buchstaben bzw. zur Darstellung der Position des Lesekopfes. Mögliche Konfigurationen sind z.B.
  - *zaabb*, *aazbb*, *aabbz#* aber auch
  - *z###aXXb*
  - *aa##zbbb*
  - *aa##bbz##cc*
  - *aabb##z#*
  - ...

Rechnung einer TM

Die Definition einer Rechnung wird dadurch verkompliziert, dass

- das (Eingabe-)Band einseitig (nach rechts) unendlich ist
- der LSK also nicht über den linken Rand bewegt werden kann
- und die Konfigurationen stets den "relevanten Teil" auf dem Eingabeband darstellen sollen (d.h. insb. das nicht alle # dargestellt werden, sondern nur jene zwischen "richtigen" Buchstaben bzw. zur Darstellung der Position des Lesekopfes. Mögliche Konfigurationen sind z.B.
  - *zaabb*, *aazbb*, *aabbz#* aber auch
  - *z###aXXb*
  - *aa##zbbb*
  - *aa##bbz##cc*
  - *aabb##z#*
  - ...

Rechnung einer TM

Die Definition einer Rechnung wird dadurch verkompliziert, dass

- das (Eingabe-)Band einseitig (nach rechts) unendlich ist
- der LSK also nicht über den linken Rand bewegt werden kann
- und die Konfigurationen stets den "relevanten Teil" auf dem Eingabeband darstellen sollen (d.h. insb. das nicht alle # dargestellt werden, sondern nur jene zwischen "richtigen" Buchstaben bzw. zur Darstellung der Position des Lesekopfes. Mögliche Konfigurationen sind z.B.
  - *zaabb*, *aazbb*, *aabbz#* aber auch
  - *z###aXXb*
  - *aa##zbbb*
  - *aa##bbz##cc*
  - *aabb##z#*
  - ...

Rechnung einer TM

Definition (Schrittrelation einer DTM)

Sei A eine DTM. Die **Schrittrelation** $\vdash_A \subseteq \text{KONF}_A \times \text{KONF}_A$ ist definiert durch: $w \vdash_A w'$ gilt gdw. 1, 2, 3 oder 4 unten gilt.
Es ist $u, v, w \in \Gamma^*$, $x, y, z \in Y$, $p, q \in Z$.

1. $w = uypxv$ und

$$w' = \begin{cases} uqyzv, \text{ falls } (v \neq \lambda \text{ oder } z \neq \#) \text{ und } \delta(p,x) = (z, L, q) \\ uqy, \text{ falls } v = \lambda, y \neq \# \text{ und } \delta(p,x) = (\#, L, q) \\ uq, \text{ falls } v = \lambda, y = \# \text{ und } \delta(p,x) = (\#, L, q) \\ uyzqv, \text{ falls } \delta(p,x) = (z, R, q) \end{cases}$$

Bemerkung

Häufig (z.B. in dem Beispiel vorhin) notiert man noch ein # wenn der Zustand ganz rechts steht. Z.B. im dritten Fall oben $uq\#$ statt uq. Entspricht nicht ganz der Definition, ist aber ok.

Rechnung einer TM

Definition (Schrittrelation einer DTM (Forts.))

Es ist $u, v, w \in \Gamma^*$, $x, y, z \in \Gamma$, $p, q \in Z$.

1. $w = uyp$ und

$$w' = \begin{cases} uqyz, \text{ falls } z \neq \# \text{ und } \delta(p, \#) = (z, L, q) \\ uqy, \text{ falls } y \neq \# \text{ und } d(p, \#) = (\#, L, q) \\ uq, \text{ falls } y = \# \text{ und } \delta(p, \#) = (\#, L, q) \\ uyzq, \text{ falls } \delta(p, \#) = (z, R, q) \end{cases}$$

2. $w = pxv$ und

$$w' = zqv, \text{ falls } \delta(p, x) = (z, R, q)$$

3. $w = p$ und

$$w' = zq, \text{ falls } \delta(p, \#) = (z, R, q)$$

Rechnung einer TM

Definition (Rechnung einer TM)

1. Mit $\vdash_A^*$ wird die reflexive, transitive Hülle von $\vdash_A$ bezeichnet. Der Index A kann auch weggelassen werden.
2. Eine Folge $k_1 \vdash k_2 \vdash \ldots \vdash k_i \vdash \ldots$ heißt **Rechnung** der TM.
3. Eine endliche Rechnung $k_1 \vdash k_2 \vdash \ldots \vdash k_n$ heißt **Erfolgsrechnung**, wenn $k_1 \in \{z_0\} \cdot \Sigma^*$ und $k_n \in \Gamma^* \cdot Z_{end} \cdot \Gamma^*$ gilt.

Rechnung einer TM

Definition (Rechnung einer TM)

1. Mit $\vdash_A^*$ wird die reflexive, transitive Hülle von $\vdash_A$ bezeichnet. Der Index A kann auch weggelassen werden.
2. Eine Folge $k_1 \vdash k_2 \vdash \ldots \vdash k_i \vdash \ldots$ heißt **Rechnung** der TM.
3. Eine endliche Rechnung $k_1 \vdash k_2 \vdash \ldots \vdash k_n$ heißt **Erfolgsrechnung**, wenn $k_1 \in \{z_0\} \cdot \Sigma^*$ und $k_n \in \Gamma^* \cdot Z_{end} \cdot \Gamma^*$ gilt.

Rechnung einer TM

Definition (Rechnung einer TM)

1. Mit $\vdash_A^*$ wird die reflexive, transitive Hülle von $\vdash_A$ bezeichnet. Der Index A kann auch weggelassen werden.
2. Eine Folge $k_1 \vdash k_2 \vdash \ldots \vdash k_i \vdash \ldots$ heißt **Rechnung** der TM.
3. Eine endliche Rechnung $k_1 \vdash k_2 \vdash \ldots \vdash k_n$ heißt **Erfolgsrechnung**, wenn $k_1 \in \{z_0\} \cdot \Sigma^*$ und $k_n \in \Gamma^* \cdot Z_{end} \cdot \Gamma^*$ gilt.

Rechnung einer TM

Eine endliche Rechnung $k_1 \vdash k_2 \vdash \ldots \vdash k_n$ heißt **Erfolgsrechnung**, wenn $k_1 \in \{z_0\} \cdot \Sigma^*$ und $k_n \in \Gamma^* \cdot Z_{end} \cdot \Gamma^*$ gilt.

Wichtige Anmerkung

Achtung: Wir starten in $z_0 w$, wobei $w \in \Sigma^*$ ein Eingabewort ist und z_0 der Startzustand. Enden tun wir in $uz_e v$, wobei z_e ein Endzustand ist und u und v irgendwelche Wörter, die aus Bandsymbolen (enthalten die Eingabesymbole) bestehen. **Insbesondere muss das Eingabewort nicht vollständig betrachtet werden!** Die TM akzeptiert, sobald sie in einen Endzustand gelangt! (Auch, wenn sie dann vielleicht noch weiterrechnen könnte!)

Rechnung einer TM

Eine endliche Rechnung $k_1 \vdash k_2 \vdash \ldots \vdash k_n$ heißt **Erfolgsrechnung**, wenn $k_1 \in \{z_0\} \cdot \Sigma^*$ und $k_n \in \Gamma^* \cdot Z_{end} \cdot \Gamma^*$ gilt.

Wichtige Anmerkung

Achtung: Wir starten in $z_0 w$, wobei $w \in \Sigma^*$ ein Eingabewort ist und z_0 der Startzustand. Enden tun wir in $uz_e v$, wobei z_e ein Endzustand ist und u und v irgendwelche Wörter, die aus Bandsymbolen (enthalten die Eingabesymbole) bestehen. **Insbesondere muss das Eingabewort nicht vollständig betrachtet werden!** Die TM akzeptiert, sobald sie in einen Endzustand gelangt! (Auch, wenn sie dann vielleicht noch weiterrechnen könnte!)

Rechnung einer TM

Eine endliche Rechnung $k_1 \vdash k_2 \vdash \ldots \vdash k_n$ heißt **Erfolgsrechnung**, wenn $k_1 \in \{z_0\} \cdot \Sigma^*$ und $k_n \in \Gamma^* \cdot Z_{end} \cdot \Gamma^*$ gilt.

Wichtige Anmerkung

Achtung: Wir starten in $z_0 w$, wobei $w \in \Sigma^*$ ein Eingabewort ist und z_0 der Startzustand. Enden tun wir in $uz_e v$, wobei z_e ein Endzustand ist und u und v irgendwelche Wörter, die aus Bandsymbolen (enthalten die Eingabesymbole) bestehen. **Insbesondere muss das Eingabewort nicht vollständig betrachtet werden!** Die TM akzeptiert, sobald sie in einen Endzustand gelangt! (Auch, wenn sie dann vielleicht noch weiterrechnen könnte!)

Akzeptierte Sprache einer TM

Definition (Akzeptierte Sprache einer TM)

Sei $A = (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$ eine DTM. Mit $L(A)$ wird die von A akzeptierte Sprache bezeichnet:

$$L(A) := \{w \in \Sigma^* \mid \exists u, v \in \Gamma^* \; \exists z_e \in Z_{end} : \; z_0 w \vdash^* u z_e v\}$$

Wichtige Anmerkung

Nochmal: Die TM akzeptiert, sobald sie einen Endzustand erreicht! Sie muss nicht das ganze Eingabewort dafür betrachten, aber sie akzeptiert das ganze Eingabewort (!), sobald sie in einen Endzustand gelangt!

Akzeptierte Sprache einer TM

Definition (Akzeptierte Sprache einer TM)

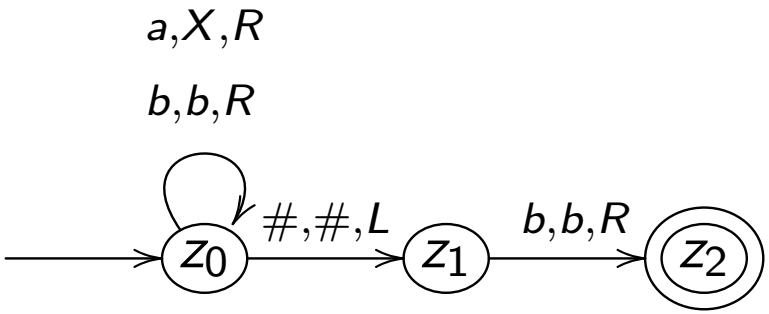
Sei $A = (Z, \Sigma, \Gamma, \delta, z_0, Z_{end})$ eine DTM. Mit $L(A)$ wird die von A akzeptierte Sprache bezeichnet:

$$L(A) := \{w \in \Sigma^* \mid \exists u, v \in \Gamma^* \; \exists z_e \in Z_{end} : \; z_0 w \vdash^* u z_e v\}$$

Wichtige Anmerkung

Nochmal: Die TM akzeptiert, sobald sie einen Endzustand erreicht! Sie muss nicht das ganze Eingabewort dafür betrachten, aber sie akzeptiert das ganze Eingabewort (!), sobald sie in einen Endzustand gelangt!

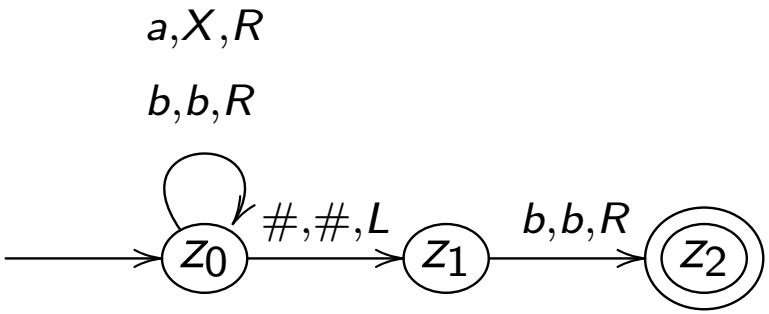
Fragen



Akzeptiert diese TM das Wort *aabba*?

1. Ja!
2. Nein!

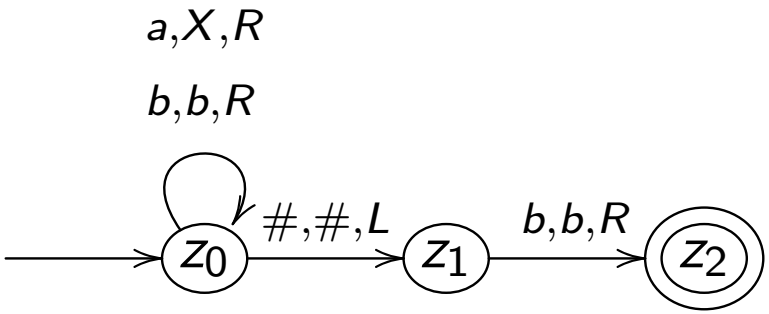
Fragen



Akzeptiert diese TM das Wort *aabb*?

1. Ja!
2. Nein!

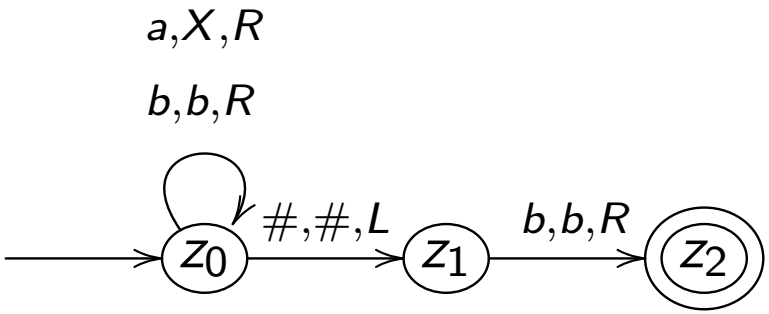
Fragen



Akzeptiert diese TM das Wort $XXbb$?

1. Ja!
2. Nein!

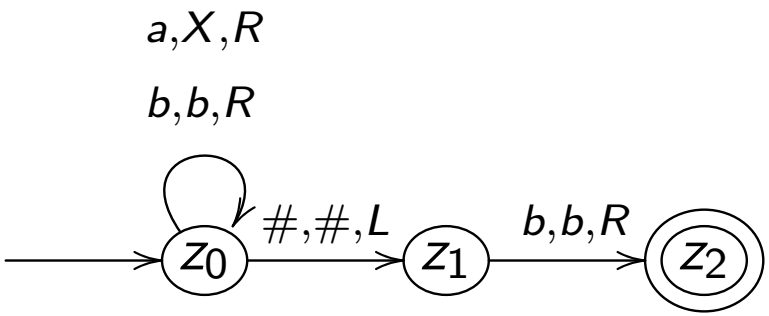
Fragen



Akzeptiert diese TM das Wort $aabb\#b$?

1. Ja!
2. Nein!

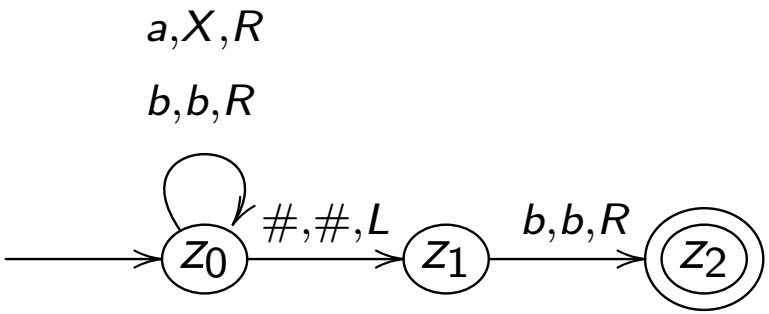
Fragen



Akzeptiert diese TM das Wort $aabb\#b$, wenn die Kante $(z_0, \#, \#, R, z_1)$ hinzugefügt wird?

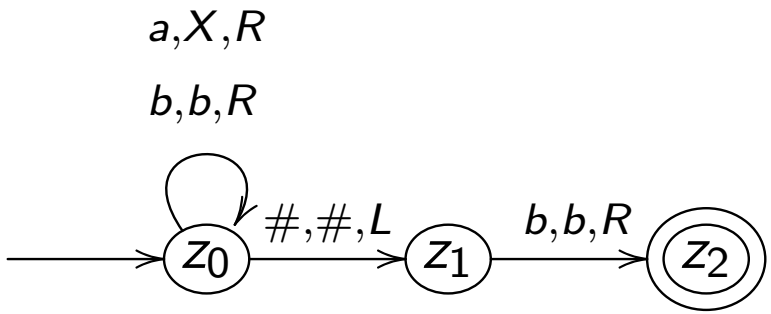
1. Ja!
2. Nein!

Fragen



Welche Sprache akzeptiert diese TM?

Fragen

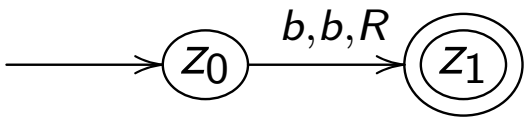


Welche Sprache akzeptiert diese TM?

Die Sprache

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

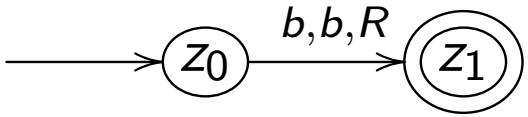
Fragen



Und welche Sprache akzeptiert diese TM?

1. $\{b\}$
2. $\{b\}^*$
3. $\{b\}^* \cdot \Sigma^*$
4. $\Sigma^* \cdot \{b\} \cdot \Sigma^*$

Fragen



Eine TM M für die Sprache

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

Fragen

Zur Nachbereitung

1. Nein!
2. Ja!
3. Nein!
4. Nein!
5. Nein!
6. Keins davon! Richtig ist $\{b\} \cdot \Sigma^*$.

Berechenbarkeit

Turing-Maschinen können nicht nur Sprachen akzeptieren, sie können auch **Funktionen berechnen**.

Berechnen von Wortfunktionen

Definition

Sei Σ ein Alphabet und $f : \Sigma^* \to \Sigma^*$ eine (möglicherweise partielle) (Wort-)Funktion. f heißt **Turing-berechenbar** oder kürzer **berechenbar** oder auch **partiell rekursiv** genau dann, wenn

es eine DTM A gibt mit $z_0 w \vdash^* z_e v$ für ein $z_e \in Z_{end}$
genau dann, wenn
$f(w) = v$ ist

Berechnen von Wortfunktionen

Definition

Sei Σ ein Alphabet und $f : \Sigma^* \to \Sigma^*$ eine (möglicherweise partielle) (Wort-)Funktion. f heißt **Turing-berechenbar** oder kürzer **berechenbar** oder auch **partiell rekursiv** genau dann, wenn

es eine DTM A gibt mit $z_0 w \vdash^* z_e v$ für ein $z_e \in Z_{end}$
genau dann, wenn
$f(w) = v$ ist

Berechnen von Funktionen

Definition

Eine (partielle) Funktion $f : \mathbb{N}^r \to \mathbb{N}^s$ heiß **(Turing-)berechenbar** oder **partiell rekursiv** genau dann, wenn es eine DTM gibt mit

$$z_0 0^{m_1+1} 1 0^{m_2+1} 1 \ldots 1 0^{m_r+1} \vdash^* z_e 0^{n_1+1} 1 0^{n_2+1} 1 \ldots 1 0^{n_s+1} \vdash^*$$

genau dann, wenn

$$f(m_1, m_2, \ldots, m_r) = (n_1, n_2, \ldots, n_s)$$

und der Funktionswert definiert ist.

Berechnen von Funktionen

Anmerkung

Oft berechnen wir aber nicht wie oben mit einer unären Kodierung, sondern mit einer binären Kodierung. Die Zahl 9 wird dann durch das **Wort** 1001 ausgedrückt. Mehrere Argumente trennen wir durch ein Symbol. So kann man z.B. eine Funktion $f(x, y) = x + y$ als Wortfunktion ausdrücken, die ein Wort wie 1001\$100 als Argument kriegt und $f(9, 4) = 13$ also das Wort 1101 berechnet.

Berechnen von Funktionen

Anmerkung

Oft berechnen wir aber nicht wie oben mit einer unären Kodierung, sondern mit einer binären Kodierung. Die Zahl 9 wird dann durch das **Wort** 1001 ausgedrückt. Mehrere Argumente trennen wir durch ein Symbol. So kann man z.B. eine Funktion $f(x, y) = x + y$ als Wortfunktion ausdrücken, die ein Wort wie 1001\$100 als Argument kriegt und $f(9, 4) = 13$ also das Wort 1101 berechnet.

Beispiel

Beispiel

Wir wollen eine TM, die $f(x) = x - 1$ berechnet. Dabei sei x eine natürliche Zahl, die in Binärkodierung gegeben ist. D.h. wir wollen eine Wortfunktion von x nach $f(x)$ berechnen.

Die TM arbeitet wie folgt:

1. Fahre zum am weitesten rechts stehenden Zeichen von x.
2. Ist dies eine 1, schreibe eine 0 und fahre fort bei Schritt 4
3. Ist dies eine 0, schreibe eine 1, gehe ein Feld nach links und mach bei Schritt 2 weiter.
4. Fahre ganz nach links und gehe in einen Endzustand.

Beispiel

Beispiel

Wir wollen eine TM, die $f(x) = x - 1$ berechnet. Dabei sei x eine natürliche Zahl, die in Binärkodierung gegeben ist. D.h. wir wollen eine Wortfunktion von x nach $f(x)$ berechnen.

Die TM arbeitet wie folgt:

1. Fahre zum am weitesten rechts stehenden Zeichen von x.
2. Ist dies eine 1, schreibe eine 0 und fahre fort bei Schritt 4
3. Ist dies eine 0, schreibe eine 1, gehe ein Feld nach links und mach bei Schritt 2 weiter.
4. Fahre ganz nach links und gehe in einen Endzustand.

Beispiel

Beispiel

Wir wollen eine TM, die $f(x) = x - 1$ berechnet. Dabei sei x eine natürliche Zahl, die in Binärkodierung gegeben ist. D.h. wir wollen eine Wortfunktion von x nach $f(x)$ berechnen.

Die TM arbeitet wie folgt:

1. Fahre zum am weitesten rechts stehenden Zeichen von x.
2. Ist dies eine 1, schreibe eine 0 und fahre fort bei Schritt 4
3. Ist dies eine 0, schreibe eine 1, gehe ein Feld nach links und mach bei Schritt 2 weiter.
4. Fahre ganz nach links und gehe in einen Endzustand.

Beispiel

Beispiel

Wir wollen eine TM, die $f(x) = x - 1$ berechnet. Dabei sei x eine natürliche Zahl, die in Binärkodierung gegeben ist. D.h. wir wollen eine Wortfunktion von x nach $f(x)$ berechnen.

Die TM arbeitet wie folgt:

1. Fahre zum am weitesten rechts stehenden Zeichen von x.
2. Ist dies eine 1, schreibe eine 0 und fahre fort bei Schritt 4
3. Ist dies eine 0, schreibe eine 1, gehe ein Feld nach links und mach bei Schritt 2 weiter.
4. Fahre ganz nach links und gehe in einen Endzustand.

Beispiel

Beispiel

Wir wollen eine TM, die $f(x) = x - 1$ berechnet. Dabei sei x eine natürliche Zahl, die in Binärkodierung gegeben ist. D.h. wir wollen eine Wortfunktion von x nach $f(x)$ berechnen.

Die TM arbeitet wie folgt:

1. Fahre zum am weitesten rechts stehenden Zeichen von x.
2. Ist dies eine 1, schreibe eine 0 und fahre fort bei Schritt 4
3. Ist dies eine 0, schreibe eine 1, gehe ein Feld nach links und mach bei Schritt 2 weiter.
4. Fahre ganz nach links und gehe in einen Endzustand.

Beispiel

Beispiel

Wir wollen eine TM, die $f(x, y) = x + y$ berechnet. Dabei seien x und y wieder natürliche Zahlen, die in Binärkodierung gegeben sind. D.h. wir wollen eine Wortfunktion von $x\$y$ nach $f(x, y)$ berechnen.

Nur mündlich ...

Berechnen und Akzeptieren

Berechnen von Funktionen und Akzeptieren von Sprachen ist ähnlich:

- Akzeptieren einer Sprache M ist Berechnen der charakteristischen Funktion von M.
- Berechnen einer Funktionen $f : M \to N$ ist Akzeptieren der Sprache $L = \{(x, f(x)) \mid x \in M\}$ (ggf. müssen M und N geeignet kodiert werden).

Bemerkung

Die charakteristische Funktion einer Menge M tut folgendes:

$$\chi_M(x) = 1 \text{ gdw. } x \in M$$

(Ist also $x \notin M$, so ist $\chi_M(x) = 0$.)

Berechnen und Akzeptieren

Berechnen von Funktionen und Akzeptieren von Sprachen ist ähnlich:

- Akzeptieren einer Sprache M ist Berechnen der charakteristischen Funktion von M.
- Berechnen einer Funktionen $f : M \to N$ ist Akzeptieren der Sprache $L = \{(x, f(x)) \mid x \in M\}$ (ggf. müssen M und N geeignet kodiert werden).

Bemerkung

Die charakteristische Funktion einer Menge M tut folgendes:

$$\chi_M(x) = 1 \text{ gdw. } x \in M$$

(Ist also $x \notin M$, so ist $\chi_M(x) = 0$.)

Berechnen und Akzeptieren

Berechnen von Funktionen und Akzeptieren von Sprachen ist ähnlich:

- Akzeptieren einer Sprache M ist Berechnen der charakteristischen Funktion von M.
- Berechnen einer Funktionen $f : M \to N$ ist Akzeptieren der Sprache $L = \{(x, f(x)) \mid x \in M\}$ (ggf. müssen M und N geeignet kodiert werden).

Bemerkung

Die charakteristische Funktion einer Menge M tut folgendes:

$$\chi_M(x) = 1 \text{ gdw. } x \in M$$

(Ist also $x \notin M$, so ist $\chi_M(x) = 0$.)

Berechnen und Akzeptieren

$$f(x, y) = x + y$$

- Berechnen der Funktion wie oben. Die TM hat dann Rechnungen der Art

$$z_0 x\$y \vdash^* z_e f(x, y)$$

- Akzeptieren der Sprache $L = \{(x, y, z) \mid x + y = z\}$, wobei x, y, z Zeichenketten aus 0 und 1 sind (Binärkodierungen natürlicher Zahlen). Besser wäre daher oben $[z]_2 = [x]_2 + [y]_2$ zu schreiben (um die Interpretation der Zeichenkette als Binärzahl deutlich zu machen).

Anmerkung

Im Grunde genommen programmieren mit einer sehr, sehr eingeschränkten Programmiersprache...

Berechnen und Akzeptieren

$$f(x, y) = x + y$$

- Berechnen der Funktion wie oben. Die TM hat dann Rechnungen der Art

$$z_0 x\$y \vdash^* z_e f(x, y)$$

- Akzeptieren der Sprache $L = \{(x, y, z) \mid x + y = z\}$, wobei x, y, z Zeichenketten aus 0 und 1 sind (Binärkodierungen natürlicher Zahlen). Besser wäre daher oben $[z]_2 = [x]_2 + [y]_2$ zu schreiben (um die Interpretation der Zeichenkette als Binärzahl deutlich zu machen).

Anmerkung

Im Grunde genommen programmieren mit einer sehr, sehr eingeschränkten Programmiersprache...

Berechnen und Akzeptieren

$$f(x, y) = x + y$$

- Berechnen der Funktion wie oben. Die TM hat dann Rechnungen der Art

$$z_0 x\$y \vdash^* z_e f(x, y)$$

- Akzeptieren der Sprache $L = \{(x, y, z) \mid x + y = z\}$, wobei x, y, z Zeichenketten aus 0 und 1 sind (Binärkodierungen natürlicher Zahlen). Besser wäre daher oben $[z]_2 = [x]_2 + [y]_2$ zu schreiben (um die Interpretation der Zeichenkette als Binärzahl deutlich zu machen).

Anmerkung

Im Grunde genommen programmieren mit einer sehr, sehr eingeschränkten Programmiersprache...

Berechnen und Akzeptieren

$$f(x, y) = x + y$$

- Berechnen der Funktion wie oben. Die TM hat dann Rechnungen der Art

$$z_0 x\$y \vdash^* z_e f(x, y)$$

- Akzeptieren der Sprache $L = \{(x, y, z) \mid x + y = z\}$, wobei x, y, z Zeichenketten aus 0 und 1 sind (Binärkodierungen natürlicher Zahlen). Besser wäre daher oben $[z]_2 = [x]_2 + [y]_2$ zu schreiben (um die Interpretation der Zeichenkette als Binärzahl deutlich zu machen).

Anmerkung

Im Grunde genommen programmieren mit einer sehr, sehr eingeschränkten Programmiersprache...

Weitere Beispiele

Beispiel 1

Eine TM für $L = \{a^n b^n \mid n \in \mathbb{N}\}$?

Idee?

Beispiel 2

Eine TM, die $f(w) = w^{rev}$ $(w \in \{0,1\}^*)$ berechnet?

Idee?

Weitere Beispiele

Beispiel 1

Eine TM für $L = \{a^n b^n \mid n \in \mathbb{N}\}$?

Idee?

Beispiel 2

Eine TM, die $f(w) = w^{rev}$ $(w \in \{0,1\}^*)$ berechnet?

Idee?

Eine TM für $f(w) = w^{rev}$

- Marker \$ sowohl links als auch rechts vom Wort w setzen.
- Lese letztes Symbol von w und ersetze es durch #.
- Wandere nach rechts über das \$ herüber und speichern das Symbol dort. (Im Zustand merken, welches Symbol zu schreiben ist.)
- Wiederhole dies, d.h. lösche das Wort w von rechts nach links, Buchstabe für Buchstabe und baue das Wort w^{rev} rechts davon und von links nach rechts Buchstabe für Buchstabe auf.
- Zu Beachten:
  - bei w muss über die schon gelöschten Buchstaben (d.h. über die #) rüber gelesen werden; bei w^{rev} muss über das schon geschriebene Teilwort herübergelesen werden.
- w wurde zu Ende gelesen, wenn man auf das zweite \$ trifft.
- Lösche die beiden \$ und bewege den Kopf an den Anfang von w^{rev}.

Eine TM für $f(w) = w^{rev}$

- Marker \$ sowohl links als auch rechts vom Wort w setzen.
- Lese letztes Symbol von w und ersetze es durch #.
- Wandere nach rechts über das \$ herüber und speichern das Symbol dort. (Im Zustand merken, welches Symbol zu schreiben ist.)
- Wiederhole dies, d.h. lösche das Wort w von rechts nach links, Buchstabe für Buchstabe und baue das Wort w^{rev} rechts davon und von links nach rechts Buchstabe für Buchstabe auf.
- Zu Beachten:
  - bei w muss über die schon gelöschten Buchstaben (d.h. über die #) rüber gelesen werden; bei w^{rev} muss über das schon geschriebene Teilwort herübergelesen werden.
- w wurde zu Ende gelesen, wenn man auf das zweite \$ trifft.
- Lösche die beiden \$ und bewege den Kopf an den Anfang von w^{rev}.

Eine TM für $f(w) = w^{rev}$

- Marker \$ sowohl links als auch rechts vom Wort w setzen.
- Lese letztes Symbol von w und ersetze es durch #.
- Wandere nach rechts über das \$ herüber und speichern das Symbol dort. (Im Zustand merken, welches Symbol zu schreiben ist.)
- Wiederhole dies, d.h. lösche das Wort w von rechts nach links, Buchstabe für Buchstabe und baue das Wort w^{rev} rechts davon und von links nach rechts Buchstabe für Buchstabe auf.
- Zu Beachten:
  - bei w muss über die schon gelöschten Buchstaben (d.h. über die #) rüber gelesen werden; bei w^{rev} muss über das schon geschriebene Teilwort herübergelesen werden.
- w wurde zu Ende gelesen, wenn man auf das zweite \$ trifft.
- Lösche die beiden \$ und bewege den Kopf an den Anfang von w^{rev}.

Eine TM für $f(w) = w^{rev}$

- Marker \$ sowohl links als auch rechts vom Wort w setzen.
- Lese letztes Symbol von w und ersetze es durch #.
- Wandere nach rechts über das \$ herüber und speichern das Symbol dort. (Im Zustand merken, welches Symbol zu schreiben ist.)
- Wiederhole dies, d.h. lösche das Wort w von rechts nach links, Buchstabe für Buchstabe und baue das Wort w^{rev} rechts davon und von links nach rechts Buchstabe für Buchstabe auf.
- Zu Beachten:
  - bei w muss über die schon gelöschten Buchstaben (d.h. über die #) rüber gelesen werden; bei w^{rev} muss über das schon geschriebene Teilwort herübergelesen werden.
- w wurde zu Ende gelesen, wenn man auf das zweite \$ trifft.
- Lösche die beiden \$ und bewege den Kopf an den Anfang von w^{rev}.

Eine TM für $f(w) = w^{rev}$

- Marker \$ sowohl links als auch rechts vom Wort w setzen.
- Lese letztes Symbol von w und ersetze es durch #.
- Wandere nach rechts über das \$ herüber und speichern das Symbol dort. (Im Zustand merken, welches Symbol zu schreiben ist.)
- Wiederhole dies, d.h. lösche das Wort w von rechts nach links, Buchstabe für Buchstabe und baue das Wort w^{rev} rechts davon und von links nach rechts Buchstabe für Buchstabe auf.
- Zu Beachten:
  - bei w muss über die schon gelöschten Buchstaben (d.h. über die #) rüber gelesen werden; bei w^{rev} muss über das schon geschriebene Teilwort herübergelesen werden.
- w wurde zu Ende gelesen, wenn man auf das zweite \$ trifft.
- Lösche die beiden \$ und bewege den Kopf an den Anfang von w^{rev}.

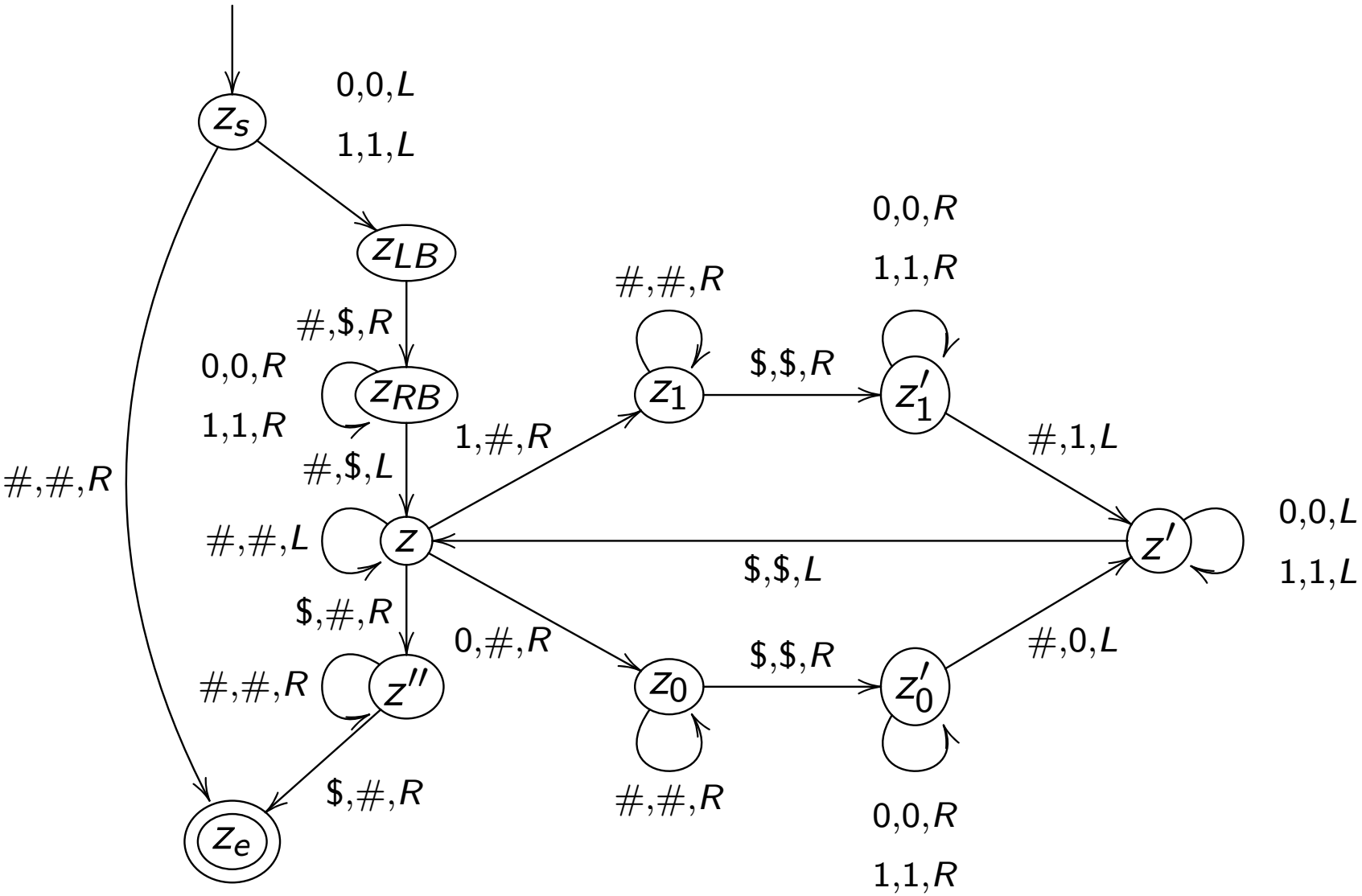
Eine TM für $f(w) = w^{rev}$

- Marker \$ sowohl links als auch rechts vom Wort w setzen.
- Lese letztes Symbol von w und ersetze es durch #.
- Wandere nach rechts über das \$ herüber und speichern das Symbol dort. (Im Zustand merken, welches Symbol zu schreiben ist.)
- Wiederhole dies, d.h. lösche das Wort w von rechts nach links, Buchstabe für Buchstabe und baue das Wort w^{rev} rechts davon und von links nach rechts Buchstabe für Buchstabe auf.
- Zu Beachten:
  - bei w muss über die schon gelöschten Buchstaben (d.h. über die #) rüber gelesen werden; bei w^{rev} muss über das schon geschriebene Teilwort herübergelesen werden.
- w wurde zu Ende gelesen, wenn man auf das zweite \$ trifft.
- Lösche die beiden \$ und bewege den Kopf an den Anfang von w^{rev}.

Eine TM für $f(w) = w^{rev}$

- Marker \$ sowohl links als auch rechts vom Wort w setzen.
- Lese letztes Symbol von w und ersetze es durch #.
- Wandere nach rechts über das \$ herüber und speichern das Symbol dort. (Im Zustand merken, welches Symbol zu schreiben ist.)
- Wiederhole dies, d.h. lösche das Wort w von rechts nach links, Buchstabe für Buchstabe und baue das Wort w^{rev} rechts davon und von links nach rechts Buchstabe für Buchstabe auf.
- Zu Beachten:
  - bei w muss über die schon gelöschten Buchstaben (d.h. über die #) rüber gelesen werden; bei w^{rev} muss über das schon geschriebene Teilwort herübergelesen werden.
- w wurde zu Ende gelesen, wenn man auf das zweite \$ trifft.
- Lösche die beiden \$ und bewege den Kopf an den Anfang von w^{rev}.

Eine TM für $f(w) = w^{rev}$



Fragen

$f : \Sigma^* \to \Sigma^*$ ist **berechenbar** gdw.
es eine DTM A gibt mit $z_0 w \vdash^* zv$
genau dann, wenn
$f(w) = v$ ist

Fehler in Zeile

1. 1
2. 2
3. 3
4. 4

Fragen

Wenn $f : \mathbb{N} \to \mathbb{N}$ und $g : \mathbb{N} \to \mathbb{N}$ Turing-berechenbar sind, dann auch $f + g$ und $f \cdot g$

1. Ja!
2. Nein!

Fragen

Wenn $f : \mathbb{N} \to \mathbb{N}$ und $g : \mathbb{N} \to \mathbb{N}$ Turing-berechenbar sind, dann auch $f \circ g$

1. Ja!
2. Nein!

Fragen

Zur Nachbereitung

1. Zweite Zeile! z muss in Z_{end} sein!
2. Ja!
3. Ja!

Varianten der TM

Definition

Zwei Turing-Maschinen A und B sind äquivalent, wenn $L(A) = L(B)$ gilt.

Varianten der TM

Definition

Die bisherige TM hat ein einseitig unendliches Band. Man kann eine TM definieren, bei der das Band in beide Richtungen (**beidseitig**) unendlich ist. Die Startkonfiguration ist dann weiterhin $z_0 w$ bei Eingabe von w, aber man kann jetzt mit dem LSK auch beliebig weit nach links wandern.

Definition

Eine k-**Band off-line Turing-Maschine** hat k beidseitig unendliche Arbeitsbänder mit jeweils einem eigenen LSK. Ferner hat die TM ein Eingabeband, auf dem sie ausschließlich lesen kann, aber den Lese-Kopf dabei in beide Richtungen bewegen darf und ein Ausgabeband, auf dem sie nur schreiben und den Schreibkopf ausschließlich von links nach rechts bewegen darf.

Varianten der TM

Definition

Die bisherige TM hat ein einseitig unendliches Band. Man kann eine TM definieren, bei der das Band in beide Richtungen (**beidseitig**) unendlich ist. Die Startkonfiguration ist dann weiterhin $z_0 w$ bei Eingabe von w, aber man kann jetzt mit dem LSK auch beliebig weit nach links wandern.

Definition

Eine k-**Band off-line Turing-Maschine** hat k beidseitig unendliche Arbeitsbänder mit jeweils einem eigenen LSK. Ferner hat die TM ein Eingabeband, auf dem sie ausschließlich lesen kann, aber den Lese-Kopf dabei in beide Richtungen bewegen darf und ein Ausgabeband, auf dem sie nur schreiben und den Schreibkopf ausschließlich von links nach rechts bewegen darf.

Varianten der TM

Satz

Zu jeder DTM A mit einseitig unendlichem Band gibt es eine äquivalente DTM B mit beidseitig unendlichem Band und umbekehrt.

Satz

Zu jeder k-Band off-line Turing-Maschine A mit $k \geq 1$ gibt es eine äquivalente DTM B mit nur einem Band.

Wichtige Anmerkung

Diese Varianten sind also äquivalent und man kann stets die TM-"Art" nehmen, die einem gerade mehr zusagt!

Literaturhinweis

Beweise zu den obigen Aussagen findet man teilweise im Skript und sonst in [HMU].

Varianten der TM

Satz

Zu jeder DTM A mit einseitig unendlichem Band gibt es eine äquivalente DTM B mit beidseitig unendlichem Band und umbekehrt.

Satz

Zu jeder k-Band off-line Turing-Maschine A mit $k \geq 1$ gibt es eine äquivalente DTM B mit nur einem Band.

Wichtige Anmerkung

Diese Varianten sind also äquivalent und man kann stets die TM-"Art" nehmen, die einem gerade mehr zusagt!

Literaturhinweis

Beweise zu den obigen Aussagen findet man teilweise im Skript und sonst in [HMU].

Nichtdeterministische Turing-Maschinen

Definition (Nichtdeterministische TM)

Eine TM $M = (Z, \Sigma, \Gamma, K, z_0, Z_{end})$ heißt **nichtdeterministische Turing-Maschine** (kurz NTM), wenn es zu jedem Paar $(z, x) \in Z \times \Gamma$ eine endliche Anzahl von Übergängen gibt. D.h. es ist

$$K \subseteq Z \times \Gamma \times \Gamma \times \{L, R, H\} \times Z$$

bzw.

$$\delta : Z \times \Gamma \to 2^{\Gamma \times \{L,R,H\} \times Z}$$

alles andere ist wie bei der DTM definiert.

Hier geht's morgen weiter ...