Berechenbarkeit und Komplexität
5. Vorlesung

TECHNISCHE UNIVERSITÄT ILMENAU

Prof. Dr. Dietrich Kuske

welt offenes Thüringen

FG Automaten und Logik, TU Ilmenau

Sommersemester 2024

Turing-Berechenbarkeit

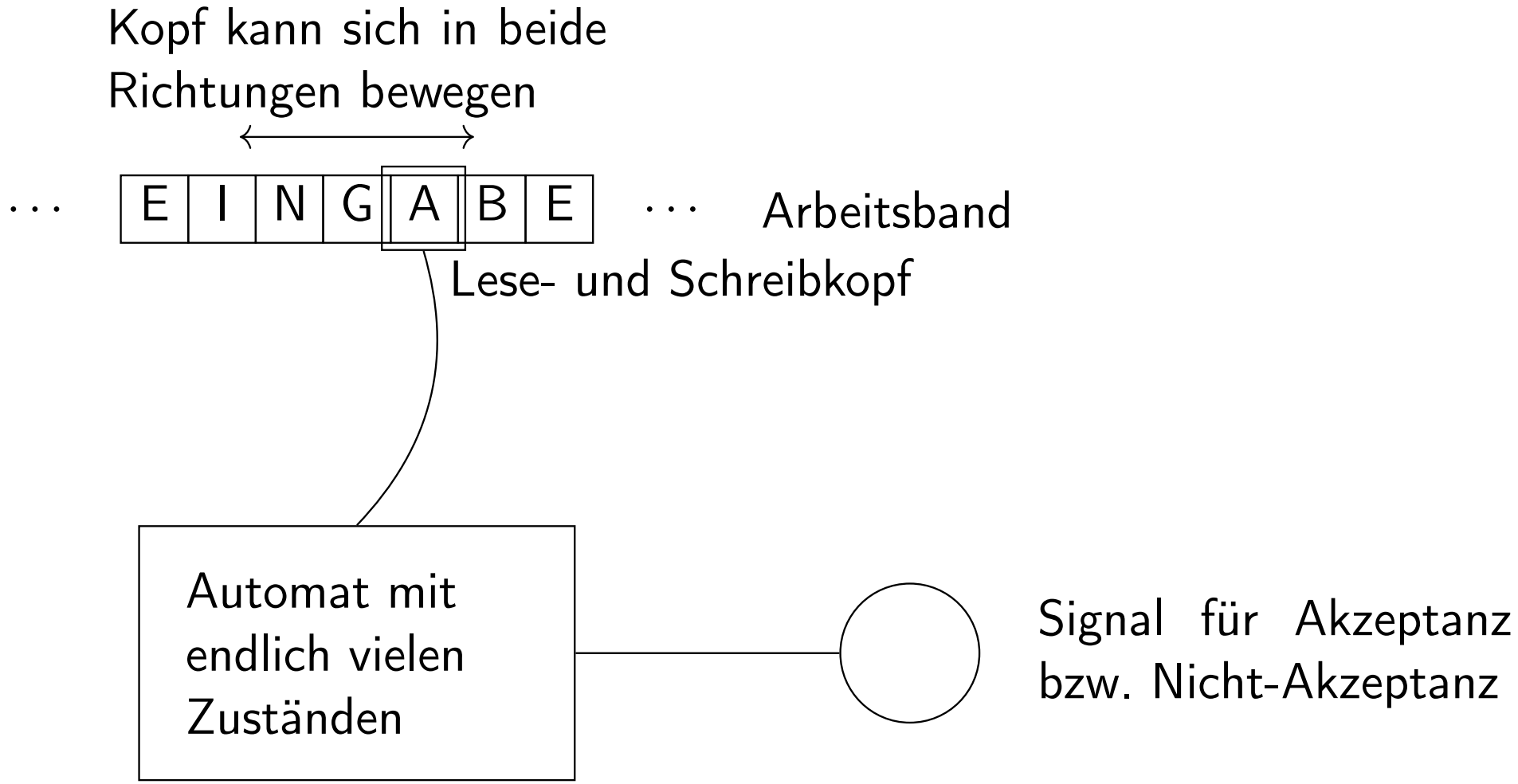
In dieser Vorlesung werden wir noch ein völlig neues Argument für die While-Vermutung behandeln: Wir werden das intuitive Berechnen analysieren, dies durch einen weiteren Algorithmenbegriff modellieren und dann zeigen, daß dieser neue Algorithmenbegriff äquivalent zur While-Berechenbarkeit ist.

Bei der Analyse des intuitiven Berechnens ging Alan Turing von den folgenden Beobachtungen aus (1936, d.h. bevor Computer gebaut wurden):

Ein „Rechnender“ kann

- sich Notizen machen auf beliebig viel Papier,
- sich endlich viel merken,
- nur einen begrenzten Teil seiner Notizen auf einmal überblicken und sich nur blätternd durch seine Notizen bewegen.
- Außerdem arbeitet er mechanisch.

Es ergibt sich das folgende Schema einer „Turingmaschine“:



Eigenschaften von Turingmaschinen:

- Wie endliche bzw. Kellerautomaten lesen Turingmaschinen eine Eingabe von einem Band und haben endlich viele Zustände.
- Im Unterschied zu endlichen und Kellerautomaten
  - kann der Lese- und Schreibkopf sich nach links und rechts bewegen,
  - Zeichen überschreiben und
  - das Band außerhalb der Eingabe verwenden.

Entsprechung „Rechnender“ $\leftrightarrow$ Turingmaschine:

Ein „Rechnender“ kann

- sich Notizen machen auf beliebig viel Papier, das linear angeordnet gedacht wird (= das Band),
- sich endlich viel merken (= endlich viele Zustände),
- nur einen begrenzten Teil seiner Notizen auf einmal überblicken (= der Lese- und Schreibkopf befindet sich immer auf genau einem Feld) und sich nur blätternd durch seine Notizen bewegen (= der Lese- und Schreibkopf bewegt sich höchstens einen Schritt nach rechts bzw. links).
- Außerdem arbeitet er mechanisch.

Ein bißchen Geschichte (vgl. Folie 1.22)

Alonzo Church und Alan Turing, die ebenso wie Hilbert und Gödel an der Frage interessiert waren, was ein „Verfahren“ sei, kannten keine While- oder Goto-Programme, sondern formulierten die folgende Vermutung:

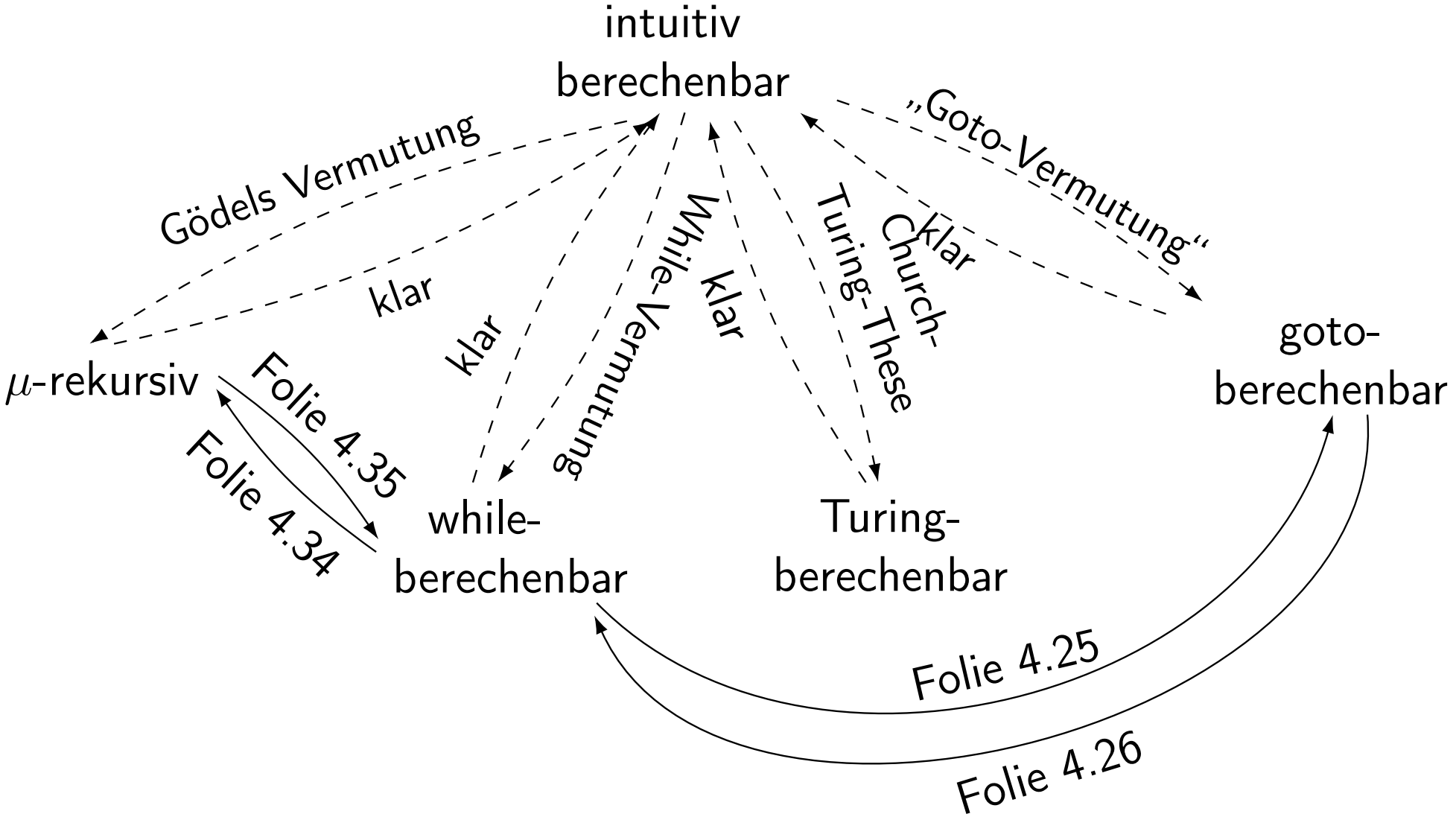
Church-Turing-These (1936)

Eine partielle Funktion $\mathbb{N}^k \dashrightarrow \mathbb{N}$ ist genau dann intuitiv berechenbar, wenn sie Turing-berechenbar ist.

genauer:

- Church vermutete, daß eine partielle Funktion genau dann intuitiv berechenbar ist, wenn sie im λ-Kalkül kodiert werden kann
- Turing zeigte, daß eine partielle Funktion genau dann im λ-Kalkül kodiert werden kann, wenn sie Turing-berechenbar ist
- Turing gab obige Argumente, warum Turing-berechenbar gleich intuitiv berechenbar ist

Zusammenfassung



Wir werden zeigen, daß die Church-Turing-These äquivalent zur While-Vermutung (Folie 4.6) und damit zu Gödels Vermutung und zur „Goto-Vermutung“ ist (für deren Gültigkeit wir ja gute Gründe haben).

Für die Gültigkeit der Church-Turing-These spricht zusätzlich, daß der Begriff „Turing-berechenbar“ aus einer Analyse der Aktivitäten eines Rechnenden abgeleitet und nicht *ad hoc* definiert ist.

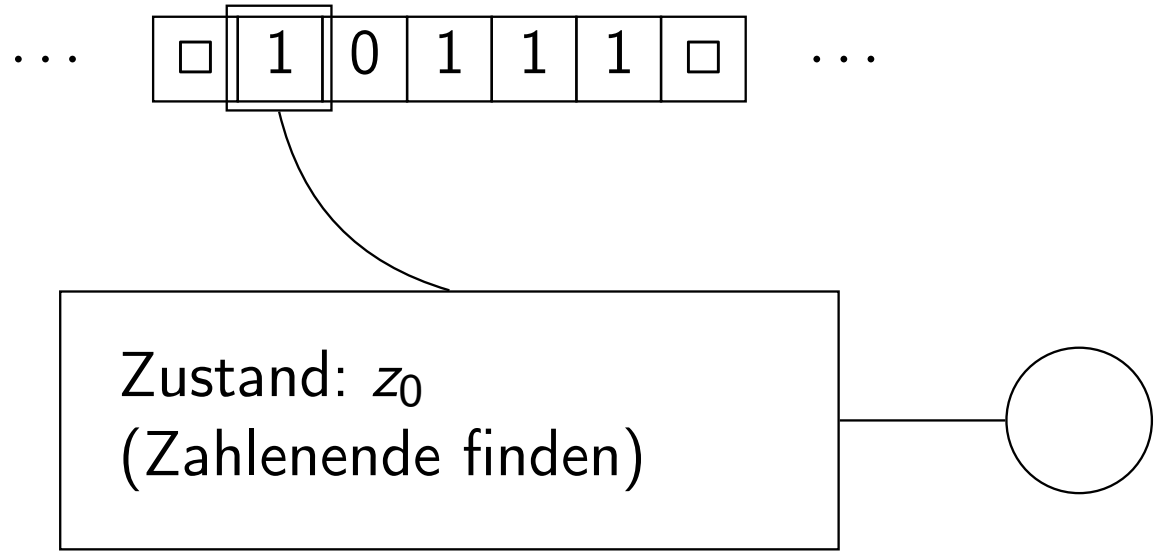
1. Beispiel einer Turingmaschine (intuitiv)

Turingmaschine, die eine Binärzahl auf dem Band um eins inkrementiert.

Idee:

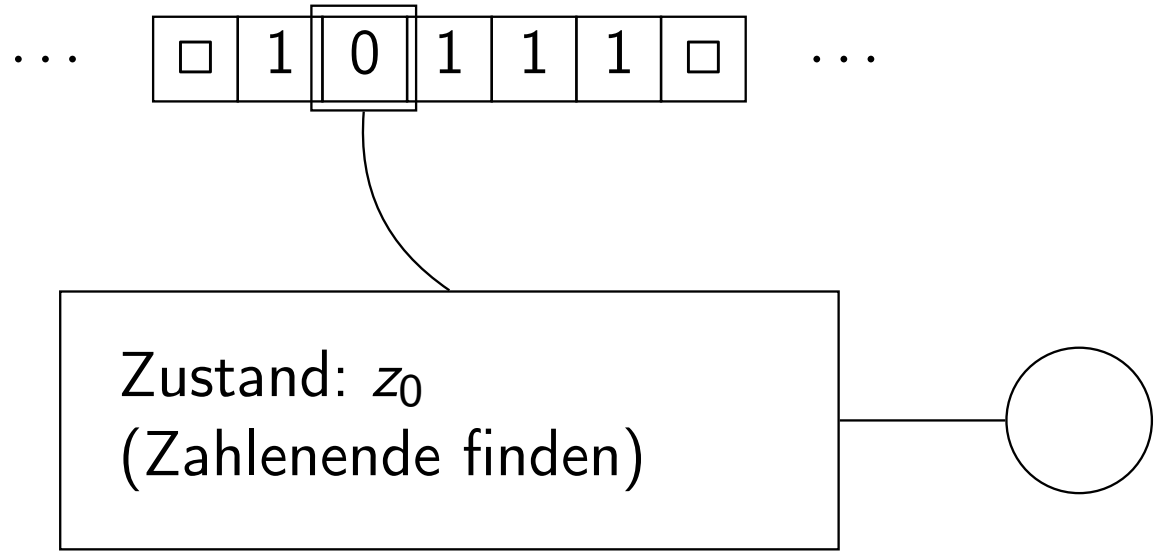
- Kopf der Turingmaschine steht zunächst auf dem am weitesten links befindlichen (höchstwertigen) Bit der Binärzahl.
- Kopf nach rechts laufen lassen, bis ein Leerzeichen gefunden wird.
- Dann wieder nach links laufen und jede 1 durch 0 ersetzen, solange bis eine 0 oder ein Leerzeichen auftaucht.
- Dieses Zeichen dann durch 1 ersetzen, bis zum Zahlanfang laufen und in einen Endzustand übergehen.

Simulation



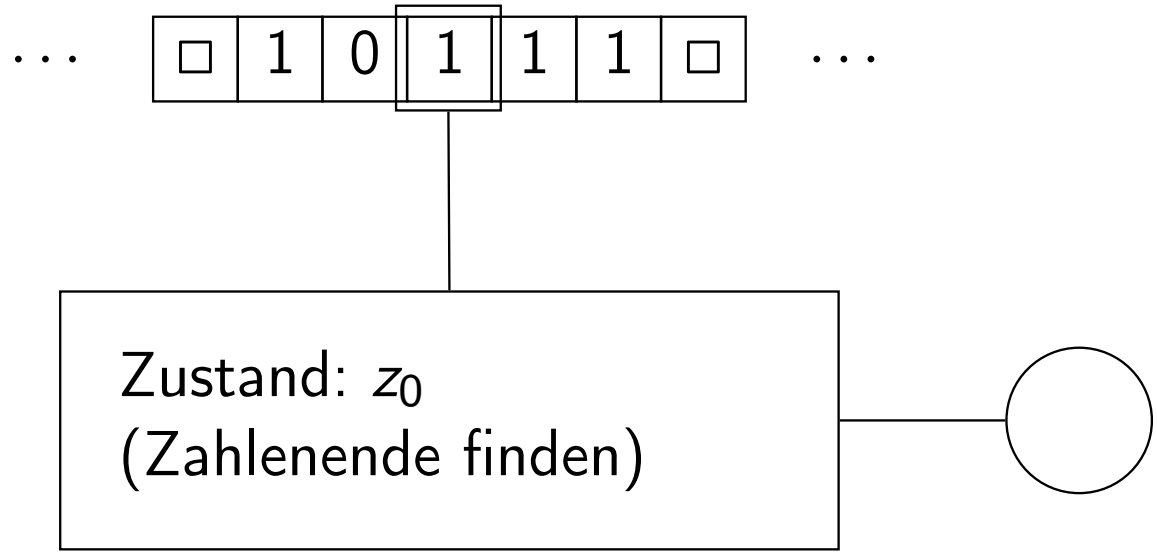
□: Leerzeichen auf dem Band

Simulation



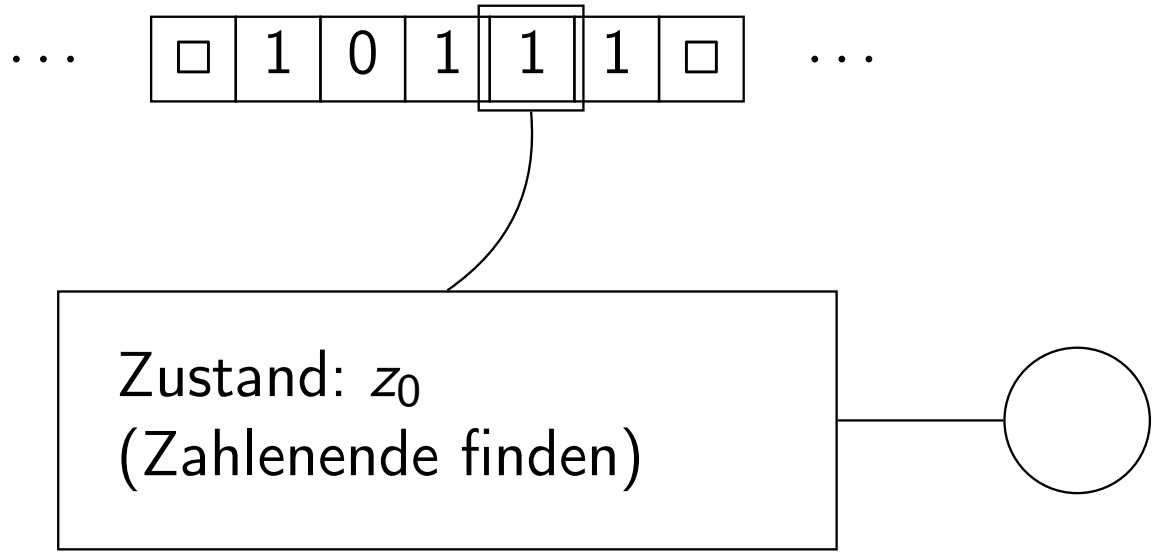
□: Leerzeichen auf dem Band

Simulation



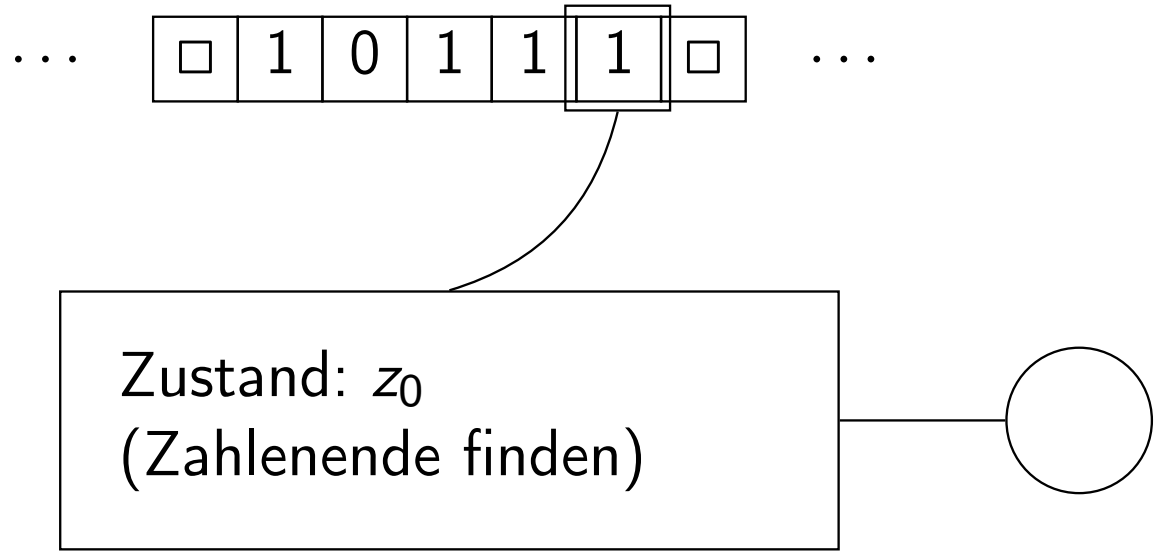
□: Leerzeichen auf dem Band

Simulation



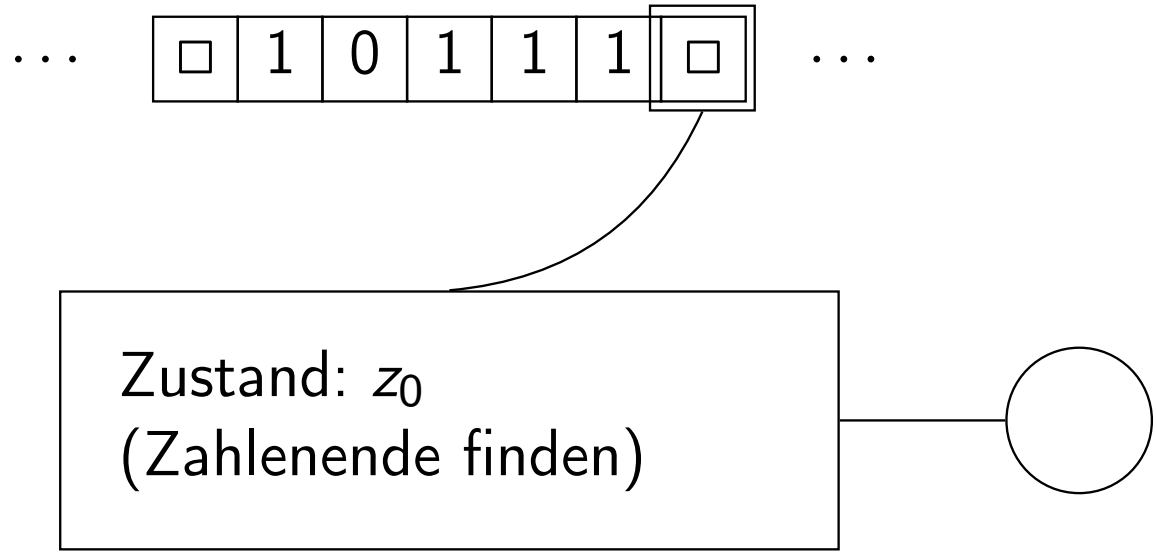
□: Leerzeichen auf dem Band

Simulation



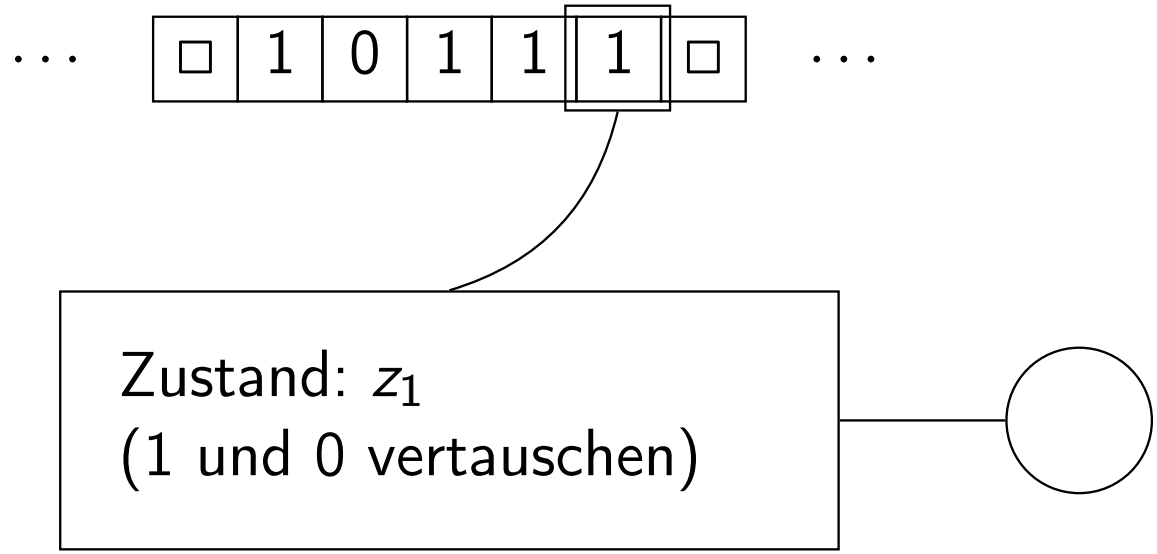
□: Leerzeichen auf dem Band

Simulation



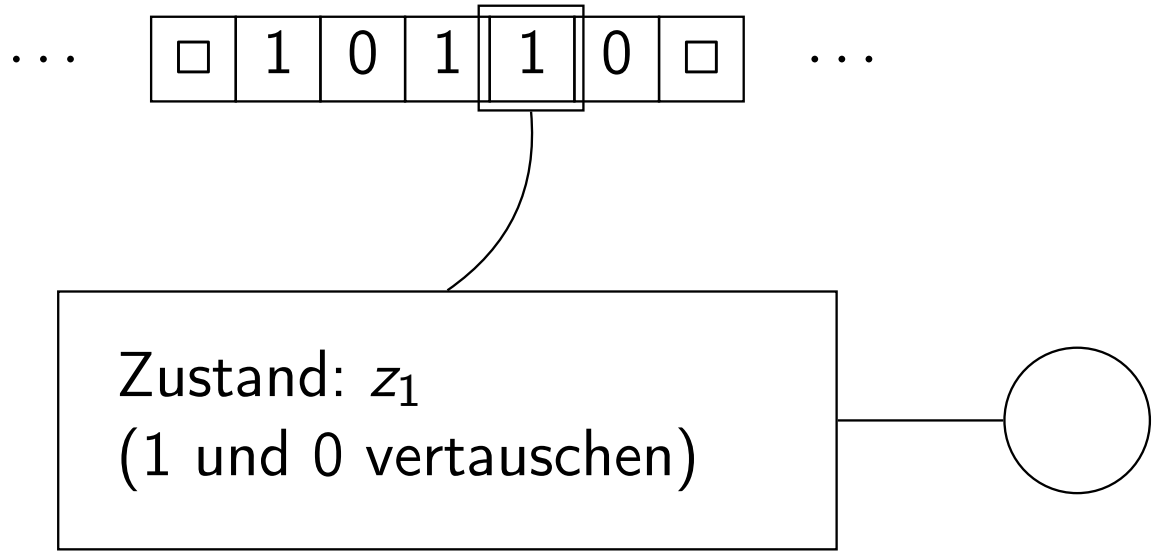
□: Leerzeichen auf dem Band

Simulation



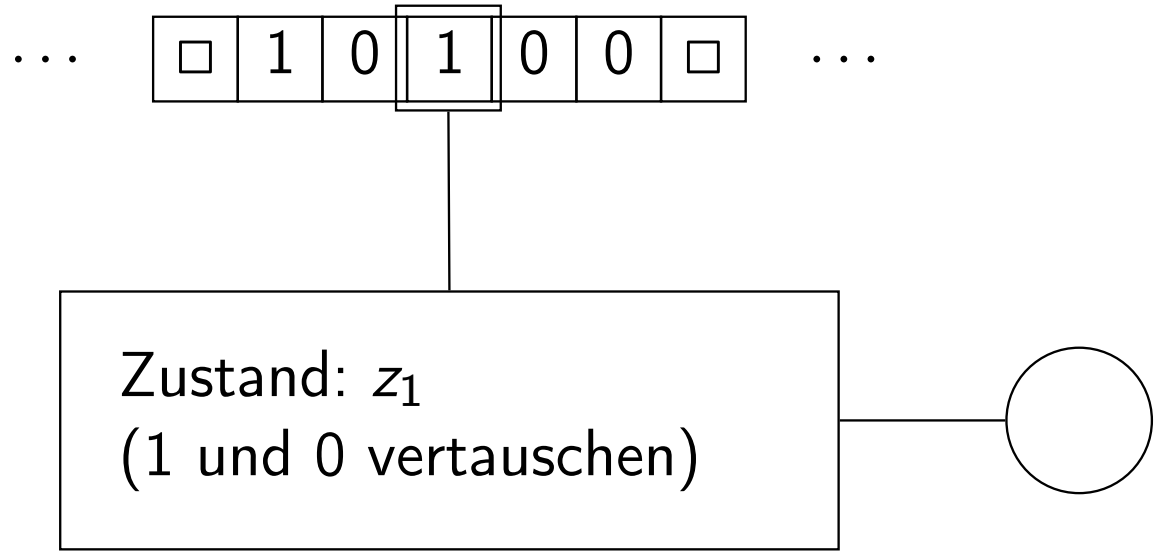
□: Leerzeichen auf dem Band

Simulation



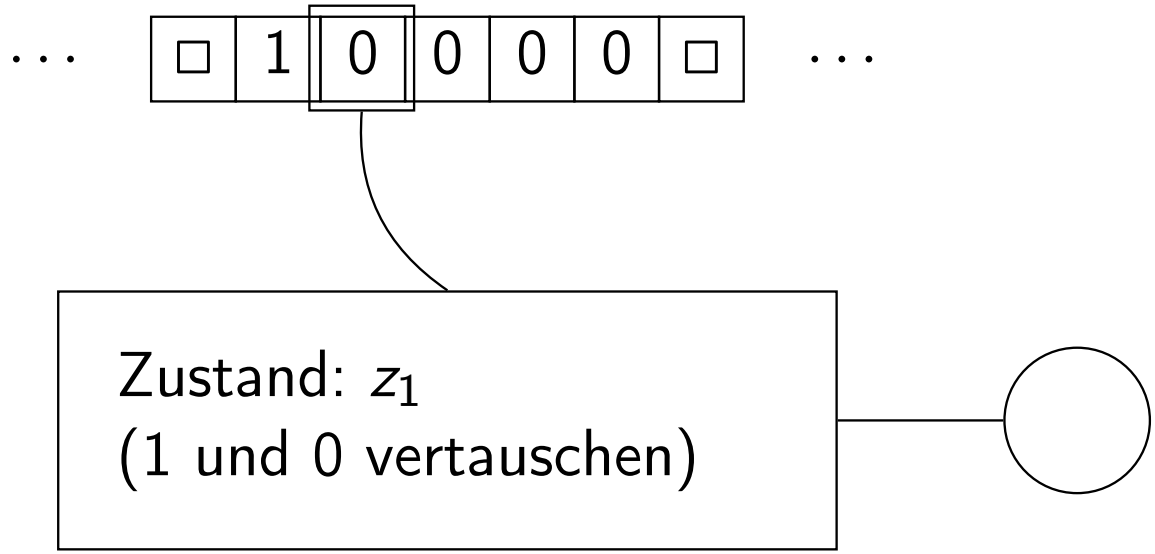
□: Leerzeichen auf dem Band

Simulation



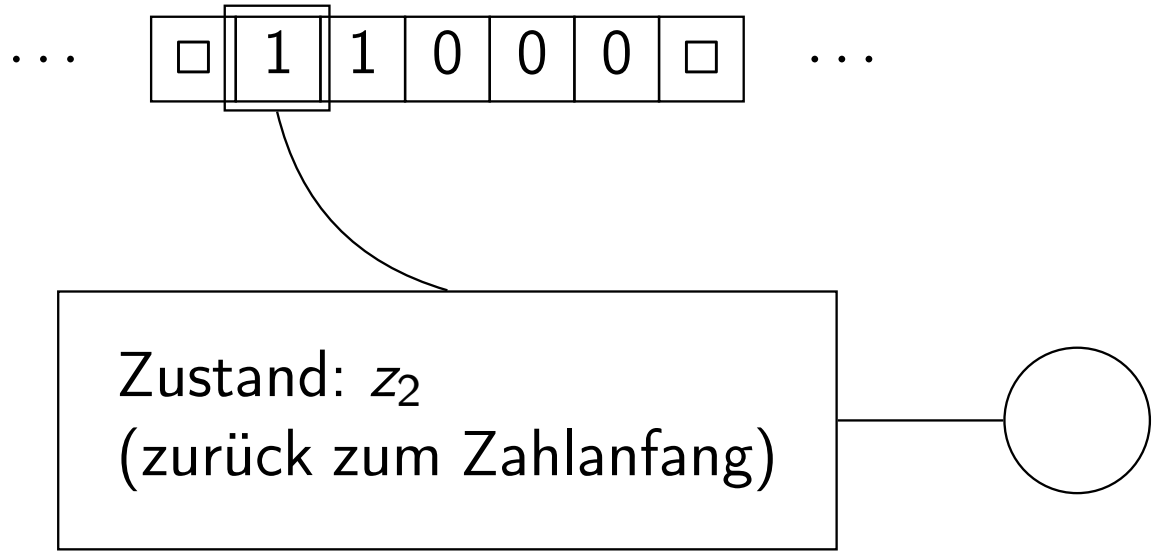
□: Leerzeichen auf dem Band

Simulation



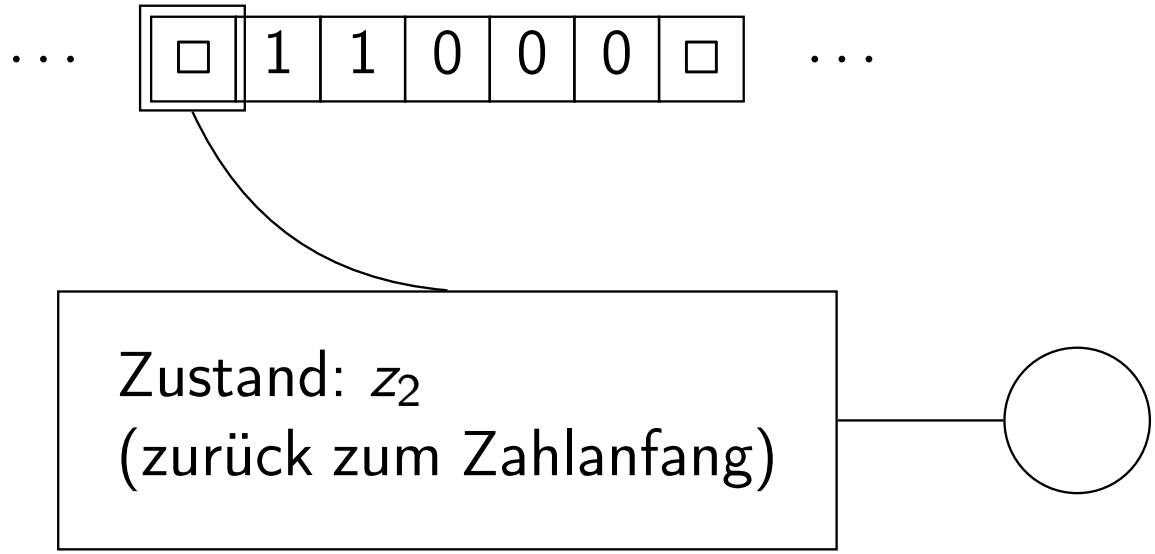
□: Leerzeichen auf dem Band

Simulation



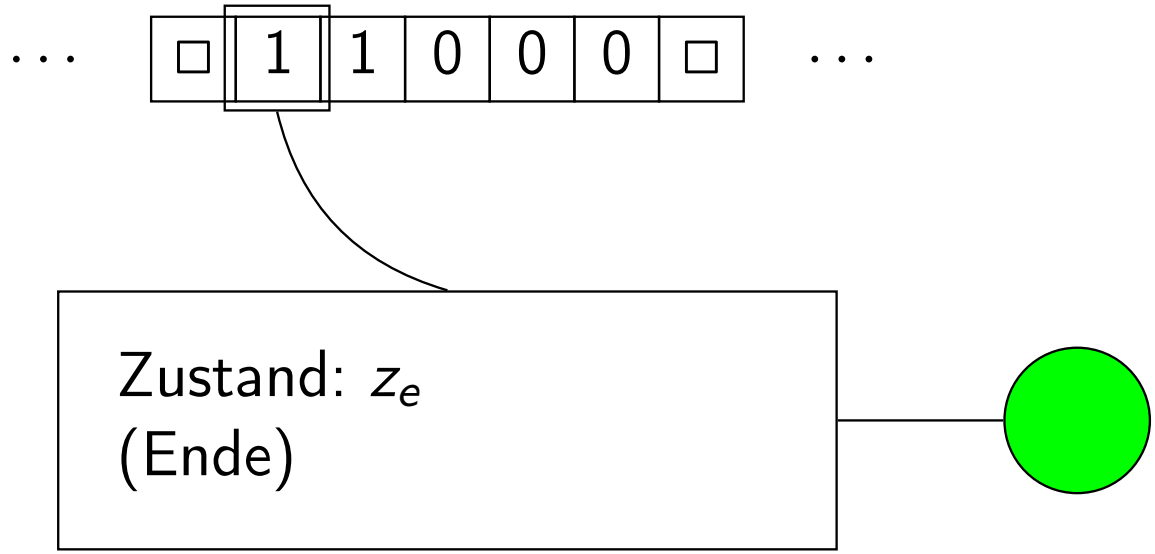
□: Leerzeichen auf dem Band

Simulation



□: Leerzeichen auf dem Band

Simulation



□: Leerzeichen auf dem Band

Turingmaschine: Definition

Definition

Eine Turingmaschine ist ein 7-Tupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$, wobei

- Z die endliche Menge der Zustände,
- Σ das Eingabealphabet,
- Γ mit $\Gamma \supseteq \Sigma$ und $\Gamma \cap Z = \varnothing$ das Arbeits- oder Bandalphabet,
- $z_0 \in Z$ der Startzustand,
- $\delta: Z \times \Gamma \to \big(Z \times \Gamma \times \{L, N, R\}\big)$ die Überführungsfunktion,
- $\square \in \Gamma \backslash \Sigma$ das Leerzeichen oder Blank und
- $E \subseteq Z$ die Menge der Endzustände ist.

Abkürzung: TM

Berechnungsschritt einer Turingmaschine: intuitiv

ein Berechnungsschritt: Falls der Lesekopf der TM im Zustand z auf dem Symbol a steht, so

(1) schlägt die TM die Anweisung $(z', b, x) = \delta(z, a)$ in ihrer „Tabelle“ nach

(2) und führt die folgenden Schritte aus:

- sie wechselt in den Zustand z',
- sie überschreibt a durch b und
- sie führt die folgende Kopfbewegung aus:
  - Kopf einen Schritt nach links, falls $x = L$,
  - Kopf bleibt stehen, falls $x = N$ und
  - Kopf einen Schritt nach rechts, falls $x = R$.

1. Beispiel einer Turingmaschine (formal)

TM zur Inkrementierung einer Binärzahl

$$M = \left(\{z_0, z_1, z_2, z_e\}, \{0,1\}, \{0,1,\square\}, \delta, z_0, \square, \{z_e\}\right) \text{ mit}$$

folgender Überführungsfunktion:

z_0: Zahlende finden
$\delta(z_0, 0) = (z_0, 0, R)$, $\delta(z_0, 1) = (z_0, 1, R)$ und $\delta(z_0, \square) = (z_1, \square, L)$

z_1: 1 und 0 vertauschen
$\delta(z_1, 0) = (z_2, 1, L)$, $\delta(z_1, 1) = (z_1, 0, L)$ und $\delta(z_1, \square) = (z_e, 1, N)$

z_2: zurück zum Zahlanfang
$\delta(z_2, 0) = (z_2, 0, L)$, $\delta(z_2, 1) = (z_2, 1, L)$ und $\delta(z_2, \square) = (z_e, \square, R)$

z_e: Endzustandsregeln
$\delta(z_e, 0) = (z_e, 0, N)$, $\delta(z_e, 1) = (z_e, 1, N)$ und $\delta(z_e, \square) = (z_e, \square, N)$

2. Beispiel einer Turingmaschine

Turingmaschine zur Berechnung einer charakteristischen Funktion

Wir suchen eine TM, die die charakteristische Funktion der (nicht kontextfreien) Sprache $L = \{0^{2^n} \mid n \geq 0\}$ berechnet.

Idee:

- Kopf steht zunächst am Beginn der Folge von Nullen.
- Links neben die Folge von Nullen 0# aufs Band schreiben.
- Wiederholt die am weitesten rechts stehende Null löschen (d.h. durch $\square$ ersetzen), nach links zum Zähler laufen und diesen um eins inkrementieren.
- Sobald alle Nullen verschwunden sind (hinter # steht $\square$), überprüfen, ob der Zähler die Form $10\cdots0$ hat.
- Zähler und # löschen, 1 bzw. 0 aufs Band schreiben und akzeptierend halten.

Berechnungsschritt einer Turingmaschine

Wie bei anderen Maschinenmodellen gibt es auch bei Turingmaschinen den Begriff einer Konfiguration, d.h., einer Momentaufnahme einer Turingmaschinen-Berechnung

Definition

Eine Konfiguration einer Turingmaschine ist ein Wort

$$k \in \Gamma^* Z \Gamma^+.$$

Bedeutung: $k = uzv$

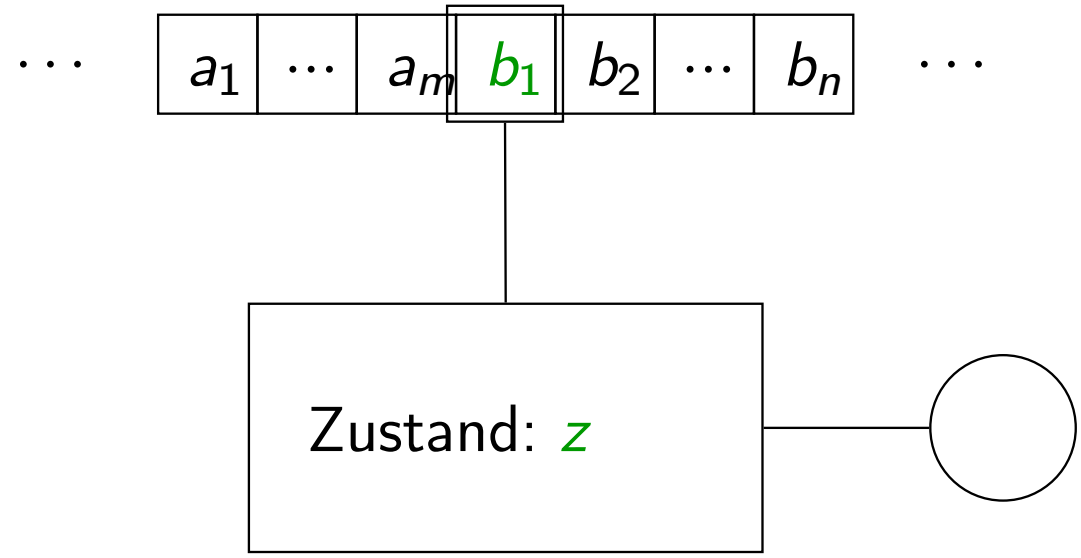
- $u \in \Gamma^*$ ist der Abschnitt des Bandes vor der Kopfposition, der bereits besucht wurde.
- $z \in Z$ ist der aktuelle Zustand.
- $v \in \Gamma^+$ ist der Abschnitt des Bandes ab der Kopfposition, der bereits besucht wurde oder im Bereich des Eingabewortes liegt. Der Kopf steht auf dem ersten Zeichen von v.

Definition einer Übergangsrelation $\vdash_M$, die beschreibt, welche Konfigurationsübergänge möglich sind.

Keine Bewegung

Es gilt: $a_1 \cdots a_m z b_1 b_2 \cdots b_n \vdash_M a_1 \cdots a_m z' c b_2 \cdots b_n$,

falls $(z', c, N) = \delta(z, b_1)$ $\quad (m \geq 0,\ n \geq 1)$.

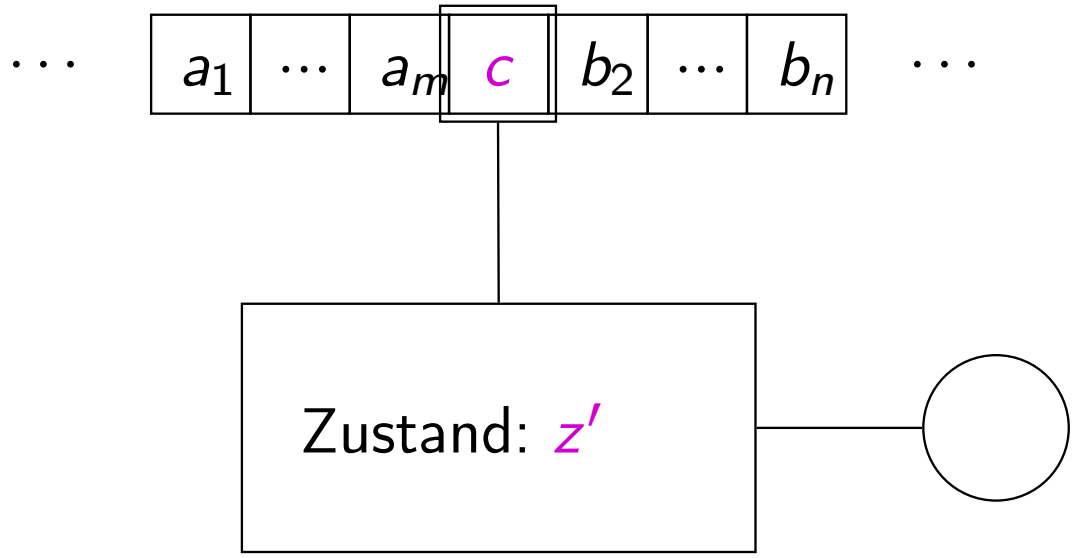


Definition einer Übergangsrelation $\vdash_M$, die beschreibt, welche Konfigurationsübergänge möglich sind.

Keine Bewegung

Es gilt: $a_1 \cdots a_m z b_1 b_2 \cdots b_n \vdash_M a_1 \cdots a_m z' c b_2 \cdots b_n$,

falls $(z', c, N) = \delta(z, b_1)$ $\quad (m \geq 0,\ n \geq 1)$.

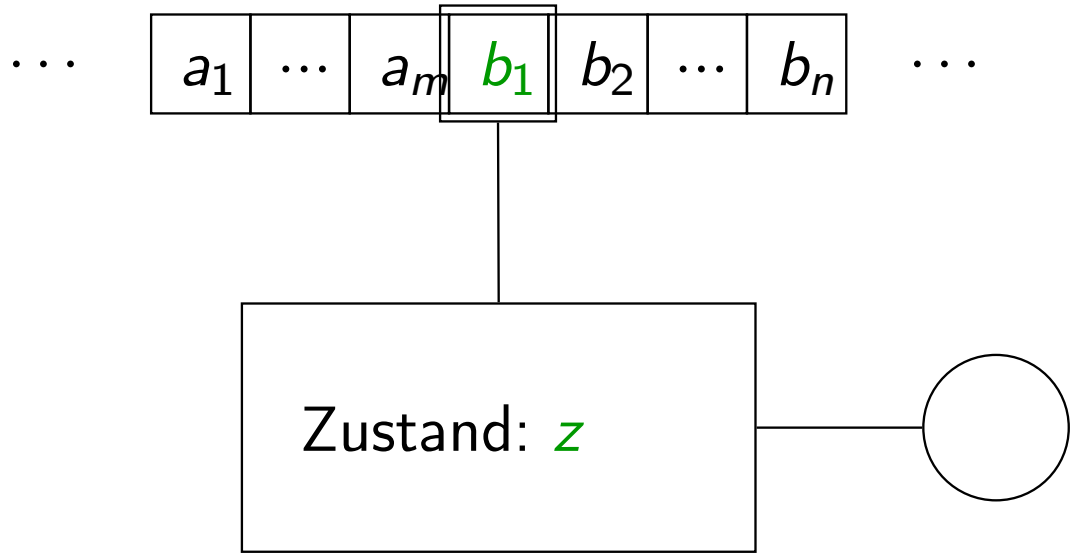


Definition einer Übergangsrelation $\vdash_M$, die beschreibt, welche Konfigurationsübergänge möglich sind.

Schritt nach links

Es gilt: $a_1 \cdots a_m z b_1 b_2 \cdots b_n \vdash_M a_1 \cdots z' a_m c b_2 \cdots b_n$,

falls $(z', c, L) = \delta(z, b_1)$ $\quad (m, n \geq 1)$.

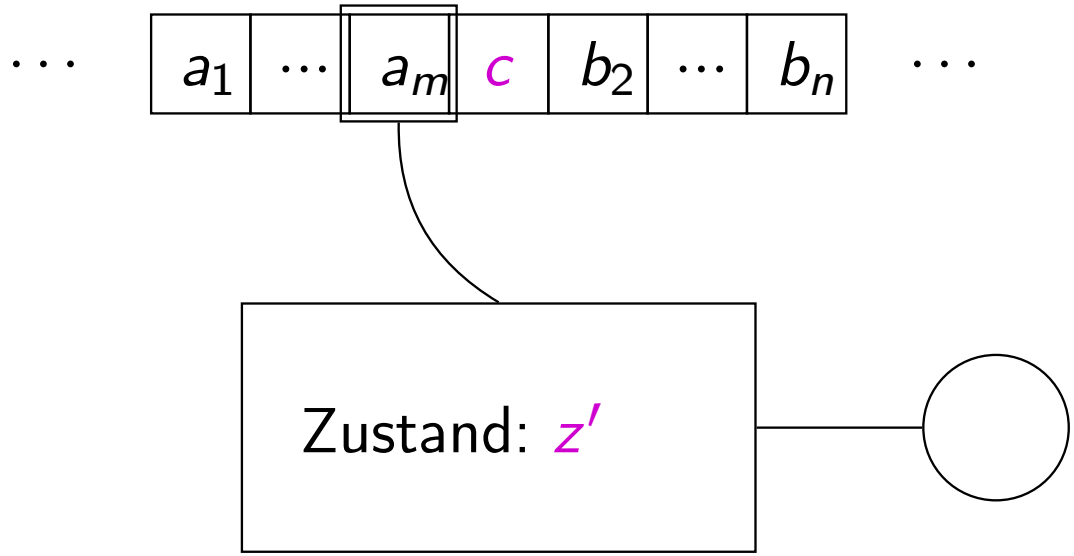


Definition einer Übergangsrelation $\vdash_M$, die beschreibt, welche Konfigurationsübergänge möglich sind.

Schritt nach links

Es gilt: $a_1 \cdots a_m z b_1 b_2 \cdots b_n \vdash_M a_1 \cdots z' a_m c b_2 \cdots b_n$,

falls $(z', c, L) = \delta(z, b_1)$ $\quad (m, n \geq 1)$.

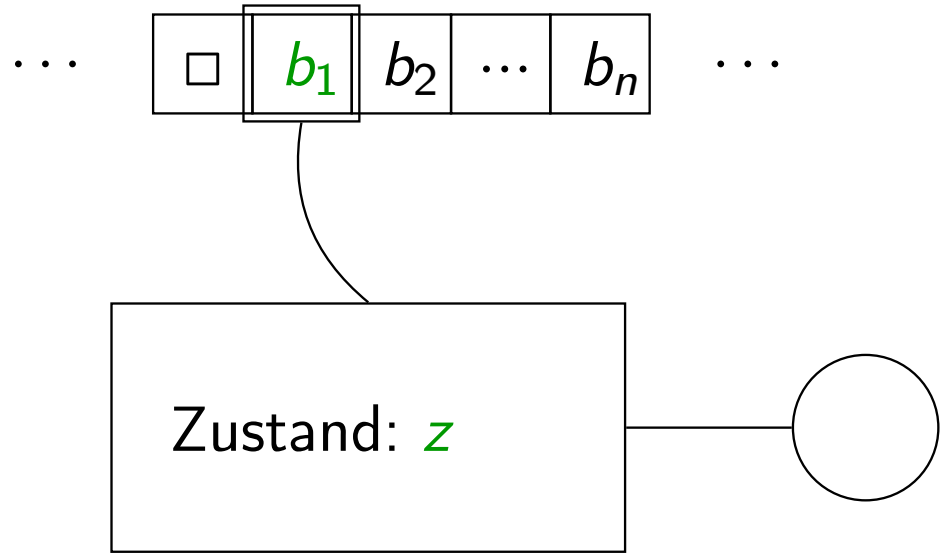


Definition einer Übergangsrelation $\vdash_M$, die beschreibt, welche Konfigurationsübergänge möglich sind.

Schritt nach links am linken Bandende

Es gilt: $z b_1 b_2 \cdots b_n \vdash_M z' \,\square\, c b_2 \cdots b_n$,

falls $(z', c, L) = \delta(z, b_1) \quad (n \geq 1)$.

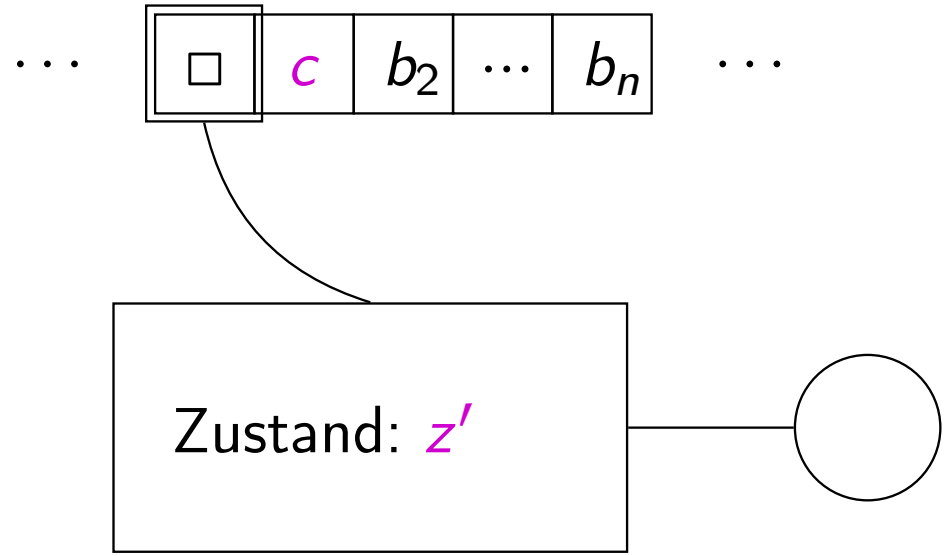


Definition einer Übergangsrelation $\vdash_M$, die beschreibt, welche Konfigurationsübergänge möglich sind.

Schritt nach links am linken Bandende

Es gilt: $z b_1 b_2 \cdots b_n \vdash_M z' \,\square\, c b_2 \cdots b_n$,

falls $(z', c, L) = \delta(z, b_1) \quad (n \geq 1)$.

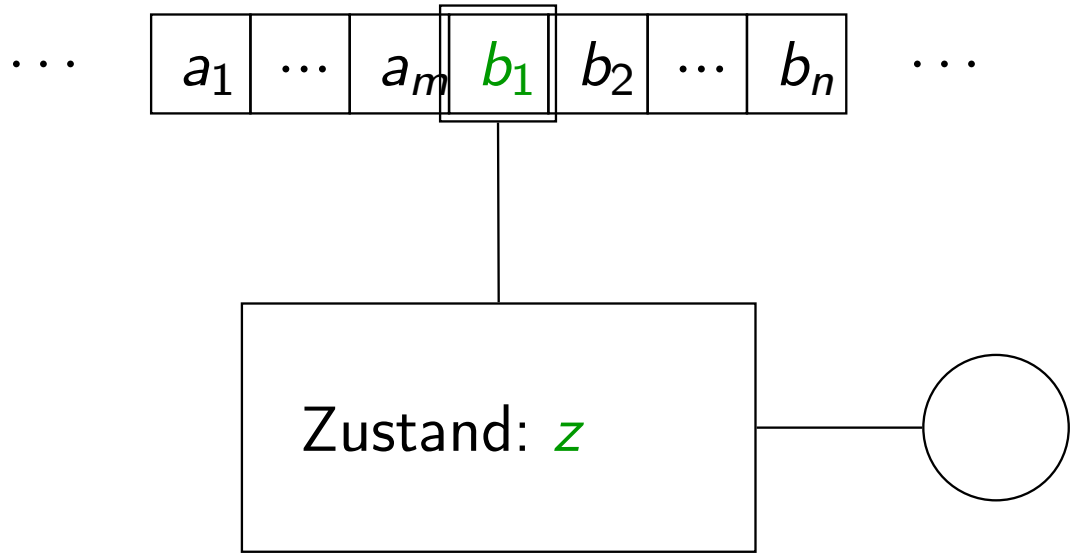


Definition einer Übergangsrelation $\vdash_M$, die beschreibt, welche Konfigurationsübergänge möglich sind.

Schritt nach rechts

Es gilt: $a_1 \cdots a_m z b_1 b_2 \cdots b_n \vdash_M a_1 \cdots a_m c z' b_2 \cdots b_n$,

falls $(z', c, R) = \delta(z, b_1)$ $\quad (m \geq 0,\ n \geq 2)$.

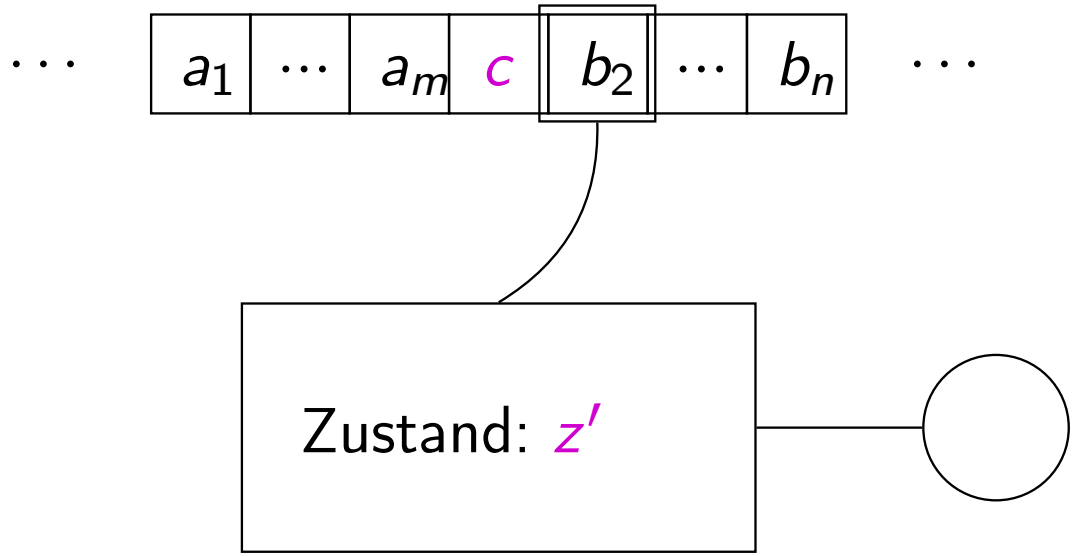


Definition einer Übergangsrelation $\vdash_M$, die beschreibt, welche Konfigurationsübergänge möglich sind.

Schritt nach rechts

Es gilt: $a_1 \cdots a_m z b_1 b_2 \cdots b_n \vdash_M a_1 \cdots a_m c z' b_2 \cdots b_n$,

falls $(z', c, R) = \delta(z, b_1)$ $\quad (m \geq 0,\ n \geq 2)$.

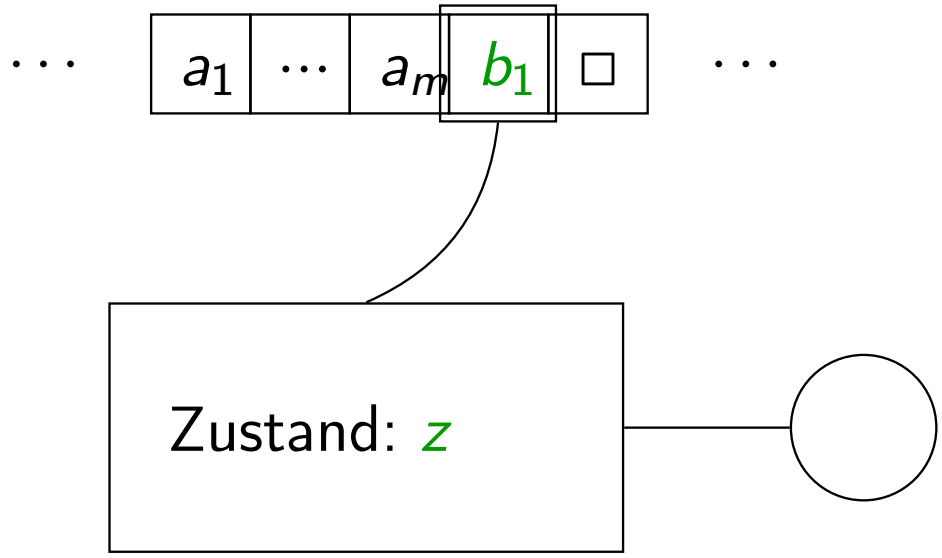


Definition einer Übergangsrelation $\vdash_M$, die beschreibt, welche Konfigurationsübergänge möglich sind.

Schritt nach rechts am rechten Bandende

Es gilt: $a_1 \dots a_m z b_1 \vdash_M a_1 \dots a_m c z' \square$,

falls $(z', c, R) = \delta(z, b_1)$ $\quad (m \geq 0)$.

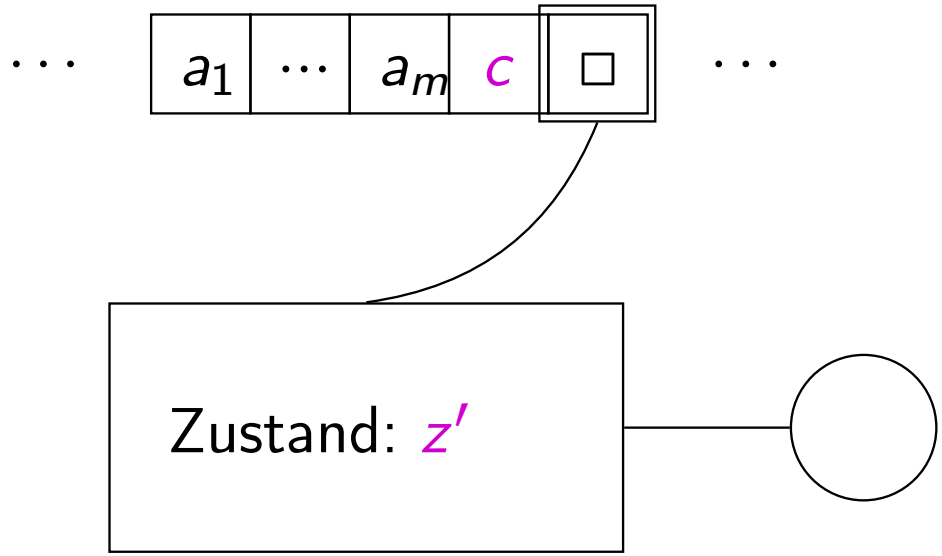


Definition einer Übergangsrelation $\vdash_M$, die beschreibt, welche Konfigurationsübergänge möglich sind.

Schritt nach rechts am rechten Bandende

Es gilt: $a_1 \dots a_m z b_1 \vdash_M a_1 \dots a_m c z' \square$,

falls $(z', c, R) = \delta(z, b_1)$ $\quad (m \geq 0)$.



Turingmaschinen: berechnete Funktion

Definition

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine TM und k eine Konfiguration. Dann heißt k Haltekonfiguration falls für alle Konfigurationen k' gilt:

$$k \vdash_M k' \Longrightarrow k = k'$$

(d.h. ist $k = uzav$, so gilt $\delta(z, a) = (z, a, N)$).

Die Haltekonfiguration k ist akzeptierend, wenn zusätzlich $k \in \square^* E \Sigma^* \square^*$ gilt.

Definition

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine TM. Die von M berechnete Funktion $f_M : \Sigma^* \dashrightarrow \Sigma^*$ ist die partielle Funktion mit (für alle $x \in \Sigma^*$):

- $f_M(x)$ ist genau dann definiert, wenn es $z_e \in E$ und $i, j \in \mathbb{N}$ gibt mit

$$z_0 x \square \vdash_M^* \square^i z_e y \square^j \text{ und } \square^i z_e y \square^j \text{ ist Haltekonfiguration}$$

- in diesem Fall gilt $f_M(x) = y$.

Intuition: wenn man das Wort x aufs Band schreibt, so berechnet die Turingmaschine daraus das Wort y (möglicherweise umgeben von Leerzeichen) und befindet sich in einer akzeptierenden Haltekonfiguration.

Definition

Eine partielle Funktion $f : \Sigma^* \dashrightarrow \Sigma^*$ heißt Turing-berechenbar, wenn es eine TM M gibt mit $f_M = f$.

Für $n \in \mathbb{N}$ sei $bin(n)$ die Binärdarstellung der Zahl n.

Definition

Sei $f: \mathbb{N}^k \dashrightarrow \mathbb{N}$ eine partielle Funktion. Definiere eine partielle Funktion $F: \{0,1,\#\}^* \dashrightarrow \{0,1,\#\}^*$ durch

$$F(w) = \begin{cases} bin\big(f(n_1,\ldots,n_k)\big) & \text{falls } w = bin(n_1)\#bin(n_2)\#\ldots\#bin(n_k) \\ & \text{und } f(n_1,\ldots,n_k) \text{ definiert ist} \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Dann heißt f Turing-berechenbar, wenn F Turing-berechenbar ist.

Bemerkung

Eine partielle Funktion $f: \mathbb{N}^k \dashrightarrow \mathbb{N}$ ist Turing-berechenbar, wenn es eine TM $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, so daß für alle $\overline{n} = (n_1, \ldots, n_k) \in \mathbb{N}^k$ gilt:

- $f_M(\overline{n})$ ist genau dann definiert, wenn es $z_e \in E$ und $i, j, m \in \mathbb{N}$ gibt mit

$$z_0 bin(n_1) \# bin(n_2) \# \cdots \# bin(n_k) \square \vdash_M^* \square^i z_e bin(m) \square^j$$

und $\square^i z_e bin(m) \square^j$ ist Haltekonfiguration

- in diesem Fall gilt $f(\overline{n}) = m$.

Intuition: Wenn man die Zahlen $n_1, \ldots, n_k$ in Binärdarstellung – voneinander getrennt durch $\#$ – aufs Band schreibt, so berechnet die Turingmaschine daraus die Binärdarstellung der Zahl $f(n_1, \ldots, n_k)$ (möglicherweise umgeben von Leerzeichen) und befindet sich in einer akzeptierenden Haltekonfiguration.

Beispiel

Die Nachfolgerfunktion $n \mapsto n+1$ ist Turing-berechenbar.

Beweis: siehe die 1. TM auf Folien 5.10 ff. □

Beispiel

Sei $\Omega: \mathbb{N} \dashrightarrow \mathbb{N}$ die überall undefinierte partielle Funktion. Diese ist Turing-berechenbar.

Beweis: Beispielsweise durch eine Turingmaschine mit der Übergangsregel

$$\delta(z_0, a) = (z_0, a, R) \qquad \text{für alle } a \in \Gamma\,.$$

□

Zusammenfassung 5. Vorlesung

in dieser Vorlesung neu

- Modellierung der Tätigkeit des Rechnens durch Turing-Maschinen

kommende Vorlesung

- Äquivalenz der Turing- und While-Berechenbarkeit, d.h. Äquivalenz von While-, Goto- und Gödels Vermutung mit der Church-Turing-These