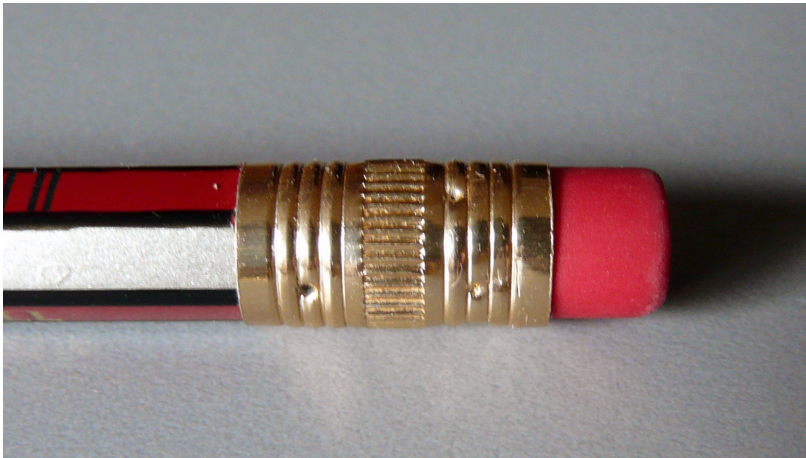


Gliederung

1. Einführung
1. Berechenbarkeitsbegriff
2. 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

Berechenbarkeitsbegriff

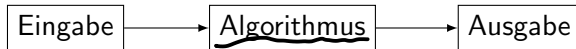


Ziel: Intuitiver Begriff \rightsquigarrow mathematische Formalisierung.

Quelle: de.wikipedia.org/wiki/Bleistift#/media/File:Bleistiftzwing_fcm.jpg

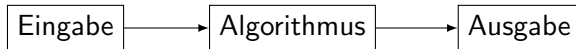
Diskussion intuitiver Berechenbarkeitsbegriff

(Intuitive) Berechenbarkeit von Funktionen:



Diskussion intuitiver Berechenbarkeitsbegriff

(Intuitive) Berechenbarkeit von Funktionen:

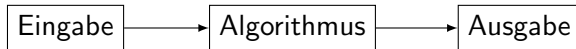


► DFA

$$\{a^n b^n \mid n \geq 0\}$$

Diskussion intuitiver Berechenbarkeitsbegriff

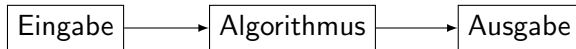
(Intuitive) Berechenbarkeit von Funktionen:



► DFA \rightsquigarrow **endlicher Speicher** nicht “berechnungsmächtig” genug

Diskussion intuitiver Berechenbarkeitsbegriff

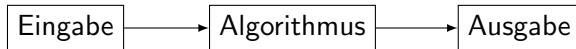
(Intuitive) Berechenbarkeit von Funktionen:



- ▶ DFA \rightsquigarrow **endlicher Speicher** nicht “berechnungsmächtig” genug
- ▶ Turing-Maschine

Diskussion intuitiver Berechenbarkeitsbegriff

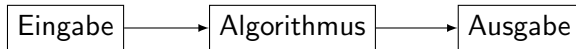
(Intuitive) Berechenbarkeit von Funktionen:



- ▶ DFA \rightsquigarrow **endlicher Speicher** nicht “berechnungsmächtig” genug
- ▶ Turing-Maschine \rightsquigarrow Speichergröße unbeschränkt

Diskussion intuitiver Berechenbarkeitsbegriff

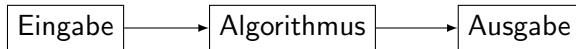
(Intuitive) Berechenbarkeit von Funktionen:



- ▶ DFA \rightsquigarrow **endlicher Speicher** nicht “berechnungsmächtig” genug
- ▶ Turing-Maschine \rightsquigarrow Speichergröße unbeschränkt
- ▶ LOOP-/WHILE-/GOTO- Programme

Diskussion intuitiver Berechenbarkeitsbegriff

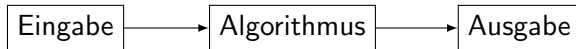
(Intuitive) Berechenbarkeit von Funktionen:



- ▶ DFA \rightsquigarrow **endlicher Speicher** nicht “berechnungsmächtig” genug
- ▶ Turing-Maschine \rightsquigarrow Speichergröße unbeschränkt
- ▶ LOOP-/WHILE-/GOTO- Programme \rightsquigarrow Variablengröße unbeschränkt

Diskussion intuitiver Berechenbarkeitsbegriff

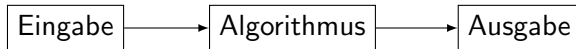
(Intuitive) Berechenbarkeit von Funktionen:



- ▶ DFA \rightsquigarrow **endlicher Speicher** nicht “berechnungsmächtig” genug
- ▶ Turing-Maschine \rightsquigarrow Speichergröße unbeschränkt
- ▶ LOOP-/WHILE-/GOTO- Programme \rightsquigarrow Variablengröße unbeschränkt
- ▶ ...

Diskussion intuitiver Berechenbarkeitsbegriff

(Intuitive) Berechenbarkeit von Funktionen:



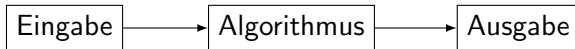
- ▶ DFA \leadsto **endlicher Speicher** nicht “berechnungsmächtig” genug
- ▶ Turing-Maschine \leadsto Speichergröße unbeschränkt
- ▶ LOOP-/WHILE-/GOTO- Programme \leadsto Variablengröße unbeschränkt
- ▶ ...

Bemerkung: leichte Diskrepanz zu modernen Computern

~~32 bit~~
~~64 bit~~

Diskussion intuitiver Berechenbarkeitsbegriff

(Intuitive) Berechenbarkeit von Funktionen:



- ▶ DFA \rightsquigarrow **endlicher Speicher** nicht “berechnungsmächtig” genug
- ▶ Turing-Maschine \rightsquigarrow Speichergröße unbeschränkt
- ▶ LOOP-/WHILE-/GOTO- Programme \rightsquigarrow Variablengröße unbeschränkt
- ▶ ...

Bemerkung: leichte Diskrepanz zu modernen Computern

Frage: Sind Turing-Maschinen mit endlichem Band genauso “mächtig” wie DFAs?

Eingabe Längen $n \rightsquigarrow \leq 2n$ Zellen ; $\leq n$?

Berechenbarkeitsbegriff

Definition

Eine (eventuell partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **berechenbar**,
wenn es einen endlichen Algorithmus \mathcal{A} gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt
$$f(n_1, \dots, n_k) = m$$



bei Eingabe (n_1, \dots, n_k) hält \mathcal{A} nach endlicher Zeit mit Ausgabe m .

Berechenbarkeitsbegriff

Definition

Eine (eventuell partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **berechenbar**,
wenn es einen **endlichen Algorithmus** \mathcal{A} gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt

$$f(n_1, \dots, n_k) = m$$


bei Eingabe (n_1, \dots, n_k) hält \mathcal{A} nach endlicher Zeit mit Ausgabe m .

Bemerkung: **existenzielle Aussage!**

Berechenbarkeitsbegriff

Definition

Eine (eventuell partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **berechenbar**, wenn es einen **endlichen Algorithmus** \mathcal{A} gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt

$$f(n_1, \dots, n_k) = m$$



bei Eingabe (n_1, \dots, n_k) hält \mathcal{A} nach endlicher Zeit mit Ausgabe m .

Bemerkung: existenzielle Aussage!

Beispiel 1 (k=1)

```
1 INPUT (n)
2 WHILE true DO { }
```

Berechenbarkeitsbegriff

Definition

Eine (eventuell partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **berechenbar**, wenn es einen **endlichen Algorithmus** \mathcal{A} gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt

$$\underline{f(n_1, \dots, n_k) = m}$$



bei Eingabe (n_1, \dots, n_k) hält \mathcal{A} nach endlicher Zeit mit Ausgabe m .

falsch



falsch

Bemerkung: existenzielle Aussage!

Beispiel 1 ($k=1$)

```
1 INPUT ( $n$ )  
2 WHILE true DO {}
```

$\leadsto \underline{\Omega}: \mathbb{N} \rightarrow \mathbb{N}$ mit $n \mapsto \perp$

Berechenbarkeitsbegriff - Beispiele

Beispiel 2

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i \in \mathbb{N}} \lfloor \pi \cdot 10^i \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Berechenbarkeitsbegriff - Beispiele

Beispiel 2

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i \in \mathbb{N}} \lfloor \pi \cdot 10^i \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

$f(n) = 1$ genau dann, wenn n genau den “ersten Dezimalstellen” von π entspricht

314 \mapsto 1
109 \mapsto 0

Berechenbarkeitsbegriff - Beispiele

Beispiel 2

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i \in \mathbb{N}} \lfloor \pi \cdot 10^i \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:



$f(n) = 1$ genau dann, wenn n genau den “ersten Dezimalstellen” von π entspricht

Algorithmus

1. approximiere π “ausreichend genau”
2. vergleiche mit Eingabe

Berechenbarkeitsbegriff - Beispiele

Beispiel 2

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i \in \mathbb{N}} \lfloor \pi \cdot 10^i \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

$f(n) = 1$ genau dann, wenn n genau den “ersten Dezimalstellen” von π entspricht



Algorithmus

1. approximiere π “ausreichend genau”
2. vergleiche mit Eingabe

Beispiel 3

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i,j,p \in \mathbb{N}} \text{ sodass } 10^j > n \text{ und} \\ & \lfloor \pi \cdot 10^i - p \cdot 10^j \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Berechenbarkeitsbegriff - Beispiele

Beispiel 2

$$f(n) := \begin{cases} 1, & \text{falls } \exists i \in \mathbb{N} \lfloor \pi \cdot 10^i \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

$f(n) = 1$ genau dann, wenn n genau den “ersten Dezimalstellen” von π entspricht



Algorithmus

1. approximiere π “ausreichend genau”
2. vergleiche mit Eingabe

Beispiel 3

$$f(n) := \begin{cases} 1, & \text{falls } \exists i, j, p \in \mathbb{N} \text{ sodass } 10^j > n \text{ und} \\ & \lfloor \pi \cdot 10^i - p \cdot 10^j \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

$f(n) = 1$ genau dann, wenn n in der Dezimalbruchentwicklung von π vorkommt

3, 14 15

$n = 41$

$n = 926$

Berechenbarkeitsbegriff - Beispiele

Beispiel 2

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i \in \mathbb{N}} \lfloor \pi \cdot 10^i \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

$f(n) = 1$ genau dann, wenn n genau den “ersten Dezimalstellen” von π entspricht



Algorithmus

1. approximiere π “ausreichend genau”
2. vergleiche mit Eingabe

Beispiel 3

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i,j,p \in \mathbb{N}} \text{ sodass } 10^j > n \text{ und} \\ & \lfloor \pi \cdot 10^i - p \cdot 10^j \rfloor = n \\ \underline{0}, & \text{sonst} \end{cases}$$

Erläuterung:

?

$f(n) = 1$ genau dann, wenn n in der Dezimalbruchentwicklung von π vorkommt

Berechenbarkeitsbegriff - Beispiele

Beispiel 2

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i \in \mathbb{N}} \lfloor \pi \cdot 10^i \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

$f(n) = 1$ genau dann, wenn n genau den “ersten Dezimalstellen” von π entspricht



Algorithmus

1. approximiere π “ausreichend genau”
2. vergleiche mit Eingabe

Beispiel 3

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i,j,p \in \mathbb{N}} \text{ sodass } 10^j > n \text{ und} \\ & \lfloor \pi \cdot 10^i - p \cdot 10^j \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:



$f(n) = 1$ genau dann, wenn n in der Dezimalbruchentwicklung von π vorkommt

Beispiel 4

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i,j,p \in \mathbb{N}} \text{ sodass } j > n \text{ und} \\ & \lfloor \pi \cdot 10^i - p \cdot 10^j \rfloor = \underbrace{11 \dots 1}_{\times n} \\ 0, & \text{sonst} \end{cases}$$

Berechenbarkeitsbegriff - Beispiele

Beispiel 2

$$f(n) := \begin{cases} 1, & \text{falls } \exists i \in \mathbb{N} \lfloor \pi \cdot 10^i \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

$f(n) = 1$ genau dann, wenn n genau den “ersten Dezimalstellen” von π entspricht



Algorithmus

1. approximiere π “ausreichend genau”
2. vergleiche mit Eingabe

Beispiel 3

$$f(n) := \begin{cases} 1, & \text{falls } \exists i, j, p \in \mathbb{N} \text{ sodass } 10^j > n \text{ und} \\ & \lfloor \pi \cdot 10^i - p \cdot 10^j \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

?

$f(n) = 1$ genau dann, wenn n in der Dezimalbruchentwicklung von π vorkommt

Beispiel 4

$$f(n) := \begin{cases} 1, & \text{falls } \exists i, j, p \in \mathbb{N} \text{ sodass } j > n \text{ und} \\ & \lfloor \pi \cdot 10^i - p \cdot 10^j \rfloor = \underbrace{11 \dots 1}_{\times n} \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

$f(n) = 1$ genau dann, wenn die Dezimalbruchentwicklung von π n konsekutive einsen enthält

Berechenbarkeitsbegriff - Beispiele

Beispiel 2

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i \in \mathbb{N}} \lfloor \pi \cdot 10^i \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

$f(n) = 1$ genau dann, wenn n genau den
"ersten Dezimalstellen" von π entspricht

Algorithmus

1. approximiere π "ausreichend genau"
2. vergleiche mit Eingabe

12 Jan in Π :

n	1	2	3	...	12	13	...
$f(n)$	1	1	1		1	1/1	

Beispiel 3

$$f(n) := \begin{cases} 1, & \text{falls } \exists_{i,j,p \in \mathbb{N}} \text{ sodass } 10^j > n \text{ und} \\ & \lfloor \pi \cdot 10^i - p \cdot 10^j \rfloor = n \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

$f(n) = 1$ genau dann, wenn n in der Dezimalbruchentwicklung von π vorkommt

Beispiel 4

$$f(n) := \begin{cases} 1, & \text{falls } \exists i, j, p \in \mathbb{N} \text{ sodass } j > n \text{ und} \\ & \lfloor \pi \cdot 10^i - p \cdot 10^j \rfloor = \underbrace{11 \dots 1}_{\times n} \\ 0, & \text{sonst} \end{cases}$$

Erläuterung:

$f(n) = 1$ genau dann, wenn die Dezimalbruchentwicklung von $\pi \cdot n$ konsequente Einsen enthält

Berechenbarkeitsbegriff II

Problem: Berechenbarkeitsbegriff basiert auf Definition von “Algorithmus”...

Berechenbarkeitsbegriff V

Problem: Berechenbarkeitsbegriff basiert auf Definition von “Algorithmus”...

Church'sche **These**

Intuitive Berechenbarkeit = Turing-Berechenbarkeit

Berechenbarkeitsbegriff V

Problem: Berechenbarkeitsbegriff basiert auf Definition von “Algorithmus”...

Church'sche **These**

Intuitive Berechenbarkeit = Turing-Berechenbarkeit

Bemerkung:

noch kein echt “mächtigeres” Berechnungsmodell als Turing-Maschine entdeckt

Berechenbarkeitsbegriff V

Problem: Berechenbarkeitsbegriff basiert auf Definition von “Algorithmus”...

Church'sche **These**

Intuitive Berechenbarkeit = Turing-Berechenbarkeit

Bemerkung:

noch kein echt “mächtigeres” Berechnungsmodell als Turing-Maschine entdeckt

Church'sche These \Rightarrow ein solches gibt es nicht

Turing-Berechenbarkeit I

Definition (Turing-Berechenbarkeit, Entscheidbarkeit)

- Eine (eventuell partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **berechenbar**, wenn es einen endlichen Algorithmus \mathcal{A} gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt $f(n_1, \dots, n_k) = m \iff$ bei Eingabe (n_1, \dots, n_k) hält \mathcal{A} nach endlicher Zeit mit Ausgabe m .

Turing-Berechenbarkeit I

Definition (Turing-Berechenbarkeit, Entscheidbarkeit)

- Eine (eventuell partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **Turing-berechenbar**, wenn es eine **D**TM $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt $f(n_1, \dots, n_k) = m \iff$ bei Eingabe (n_1, \dots, n_k) hält M nach endlicher Zeit mit Ausgabe m .

Turing-Berechenbarkeit I

Definition (Turing-Berechenbarkeit, Entscheidbarkeit)

- Eine (eventuell partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **Turing-berechenbar**, wenn es eine **DTM** $M = (Z, \Sigma, \Gamma, \delta, \underline{z_0}, \square, E)$ gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt
$$f(n_1, \dots, n_k) = m \iff \text{bei Eingabe } (n_1, \dots, n_k) \text{ hält } M \text{ nach endlicher Zeit mit Ausgabe } m.$$
$$\iff \exists_{z \in E} \underline{z_0 \text{ BIN}(n_1) \# \dots \# \text{BIN}(n_k)} \vdash_M^* \underline{z \text{ BIN}(m)} \quad \# \in \Sigma$$
wobei **BIN** Zahlen auf ihre Binärdarstellung abbildet.

Turing-Berechenbarkeit I

Definition (Turing-Berechenbarkeit, Entscheidbarkeit)

- Eine (eventuell partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **Turing-berechenbar**, wenn es eine **DTM** $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt
 $f(n_1, \dots, n_k) = m \iff$ bei Eingabe (n_1, \dots, n_k) hält M nach endlicher Zeit mit Ausgabe m .

$$\iff \exists_{z \in E} z_0 \text{ BIN}(n_1) \# \dots \# \text{BIN}(n_k) \vdash_M^* z \text{ BIN}(m)$$

wobei **BIN** Zahlen auf ihre Binärdarstellung abbildet.

- Eine (eventuell partielle) Funktion $f: \underline{\Sigma}^* \rightarrow \Sigma^*$ heißt **Turing-berechenbar**, wenn es eine **DTM** $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, sodass für alle $x, y \in \Sigma^*$ gilt

$$\underline{f(x)} = \underline{y} \iff \exists_{z \in E} \underline{z_0 x} \vdash_M^* \underline{zy}$$

Turing-Berechenbarkeit I

Definition (Turing-Berechenbarkeit, Entscheidbarkeit)

- Eine (eventuell partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **Turing-berechenbar**, wenn es eine **DTM** $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt
$$f(n_1, \dots, n_k) = m \iff \text{bei Eingabe } (n_1, \dots, n_k) \text{ hält } M \text{ nach endlicher Zeit mit Ausgabe } m.$$
$$\iff \exists_{z \in E} z_0 \text{ BIN}(n_1) \# \dots \# \text{ BIN}(n_k) \vdash_M^* z \text{ BIN}(m)$$

wobei **BIN** Zahlen auf ihre Binärdarstellung abbildet.

- Eine (eventuell partielle) Funktion $f: \Sigma^* \rightarrow \Sigma^*$ heißt **Turing-berechenbar**, wenn es eine **DTM** $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, sodass für alle $x, y \in \Sigma^*$ gilt

$$f(x) = y \iff \exists_{z \in E} z_0 x \vdash_M^* z y$$

- Eine **Sprache** L heißt **entscheidbar** wenn χ_L berechenbar ist und **semi-entscheidbar** wenn χ'_L berechenbar ist

Turing-Berechenbarkeit I

Definition (Turing-Berechenbarkeit, Entscheidbarkeit)

- Eine (eventuell partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **Turing-berechenbar**, wenn es eine **DTM** $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt
$$f(n_1, \dots, n_k) = m \iff \text{bei Eingabe } (n_1, \dots, n_k) \text{ hält } M \text{ nach endlicher Zeit mit Ausgabe } m.$$
$$\iff \exists_{z \in E} z_0 \text{ BIN}(n_1) \# \dots \# \text{ BIN}(n_k) \vdash_M^* z \text{ BIN}(m)$$


wobei **BIN** Zahlen auf ihre Binärdarstellung abbildet.

- Eine (eventuell partielle) Funktion $f: \Sigma^* \rightarrow \Sigma^*$ heißt **Turing-berechenbar**, wenn es eine **DTM** $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, sodass für alle $x, y \in \Sigma^*$ gilt

$$f(x) = y \iff \exists_{z \in E} z_0 x \vdash_M^* z y$$

- Eine **Sprache** L heißt **entscheidbar** wenn χ_L berechenbar ist und **semi-entscheidbar** wenn χ'_L berechenbar ist

Charakteristische Funktion


$$\chi_L(x) = \begin{cases} 1, & \text{falls } x \in L \\ 0, & \text{falls } x \notin L \end{cases}$$

Halbe Charakteristische Fkt.

$$\chi'_L(x) = \begin{cases} 1, & \text{falls } x \in L \\ \perp, & \text{falls } x \notin L \end{cases}$$

Turing-Berechenbarkeit I

Definition (Turing-Berechenbarkeit, Entscheidbarkeit)

- Eine (eventuell partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **Turing-berechenbar**, wenn es eine **DTM** $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, sodass für alle $n_1, \dots, n_k, m \in \mathbb{N}$ gilt
$$f(n_1, \dots, n_k) = m \iff \text{bei Eingabe } (n_1, \dots, n_k) \text{ hält } M \text{ nach endlicher Zeit mit Ausgabe } m.$$
$$\iff \exists_{z \in E} z_0 \text{ BIN}(n_1) \# \dots \# \text{ BIN}(n_k) \vdash_M^* z \text{ BIN}(m)$$

wobei **BIN** Zahlen auf ihre Binärdarstellung abbildet.

- Eine (eventuell partielle) Funktion $f: \Sigma^* \rightarrow \Sigma^*$ heißt **Turing-berechenbar**, wenn es eine **DTM** $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, sodass für alle $x, y \in \Sigma^*$ gilt

$$f(x) = y \iff \exists_{z \in E} z_0 x \vdash_M^* z y$$

- Eine **Sprache** L heißt
entscheidbar wenn χ_L berechenbar ist und
semi-entscheidbar wenn χ'_L berechenbar ist

Charakteristische Funktion

$$\chi_L(x) = \begin{cases} 1, & \text{falls } x \in L \\ 0, & \text{falls } x \notin L \end{cases}$$

Halbe Charakteristische Fkt.

$$\chi'_L(x) = \begin{cases} 1, & \text{falls } x \in L \\ \perp, & \text{falls } x \notin L \end{cases}$$

Frage: Wie hängen "Akzeptanz" und "(Semi-)Entscheidbarkeit" zusammen?

Turing-Berechenbarkeit - Beispiele

Turing-berechenbar?

Turing-Berechenbarkeit - Beispiele

Turing-berechenbar?

1. Nachfolgerfunktion $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}, n \mapsto n + 1$

Turing-Berechenbarkeit - Beispiele

Turing-berechenbar?

1. Nachfolgerfunktion $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}, n \mapsto n + 1$



Turing-Berechenbarkeit - Beispiele

Turing-berechenbar?

1. Nachfolgerfunktion $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}, n \mapsto n + 1$
2. nirgends definierte „Funktion“ Ω



Turing-Berechenbarkeit - Beispiele

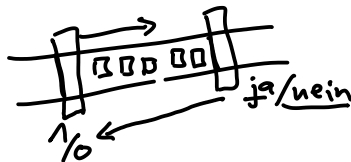
Turing-berechenbar?

1. Nachfolgerfunktion $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}, n \mapsto n + 1$ ✓
2. nirgends definierte „Funktion“ Ω ✓

Turing-Berechenbarkeit - Beispiele

Turing-berechenbar?

1. Nachfolgerfunktion $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}, n \mapsto n + 1$ ✓
2. nirgends definierte „Funktion“ Ω ✓
3. χ_L für L vom Typ 3?



Turing-Berechenbarkeit - Beispiele

Turing-berechenbar?

1. Nachfolgerfunktion $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}, n \mapsto n + 1$ ✓
2. nirgends definierte „Funktion“ Ω ✓
3. χ_L für L vom Typ 3? ✓

Turing-Berechenbarkeit - Beispiele

Turing-berechenbar?

1. Nachfolgerfunktion $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}, n \mapsto n + 1$ ✓
2. nirgends definierte „Funktion“ Ω ✓
3. χ_L für L vom Typ 3? ✓
4. χ_L für $L = \{0^n 1^n \mid n \in \mathbb{N}\}$?

Turing-Berechenbarkeit - Beispiele

Turing-berechenbar?

1. Nachfolgerfunktion $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}, n \mapsto n + 1$ ✓
2. nirgends definierte „Funktion“ Ω ✓
3. χ_L für L vom Typ 3? ✓
4. χ_L für $L = \{0^n 1^n \mid n \in \mathbb{N}\}$? ✓

Turing-Berechenbarkeit - Beispiele

Turing-berechenbar?

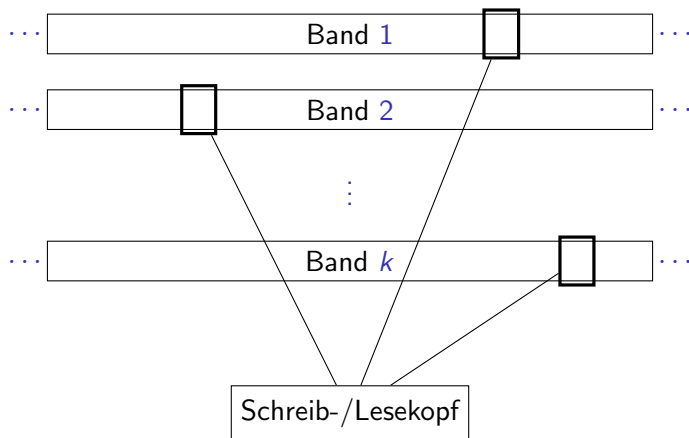
1. Nachfolgerfunktion $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}, n \mapsto n + 1$ ✓
2. nirgends definierte „Funktion“ Ω ✓
3. χ_L für L vom Typ 3? ✓
4. χ_L für $L = \{0^n 1^n \mid n \in \mathbb{N}\}$? ✓

Turing

Frage: Entspricht unsere TM für 4. **genau** der Berechenbarkeitsdefinition?

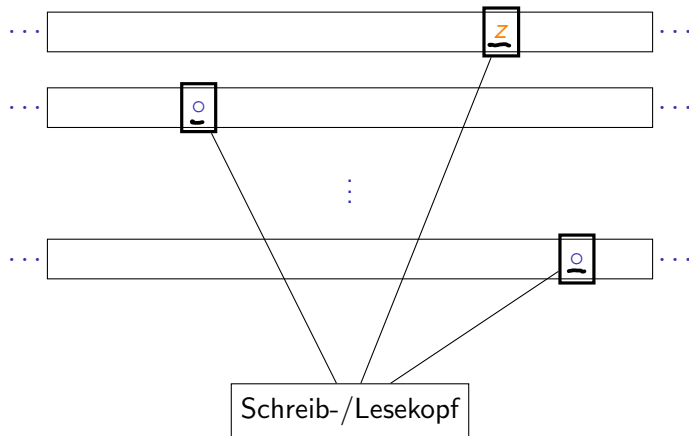
Mehrband-Turing-Maschinen I

...erlauben bequemes Programmieren (um Berechenbarkeit zu zeigen)



Mehrband-Turing-Maschinen I

...erlauben bequemes Programmieren (um Berechenbarkeit zu zeigen)



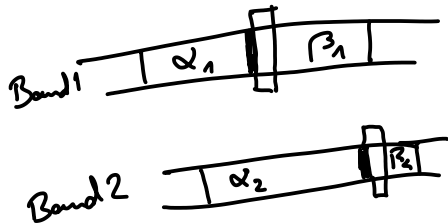
(Deterministische) k -Band Turing-Maschine M

Überföhrungsfunktion $\delta: \underline{(Z \setminus E)} \times \underline{\Gamma^k} \rightarrow \underline{Z} \times (\underline{\Gamma} \times \{\underline{L, R, N}\})^{\textcircled{k}}$.

(Deterministische) k -Band Turing-Maschine M

Überföhrungsfunktion $\delta: (Z \setminus E) \times \Gamma^k \rightarrow Z \times (\Gamma \times \{L, R, N\})^k$.

Konfiguration $\alpha_1 \underline{z} \beta_1, \alpha_2 \underline{z} \beta_2, \dots, \alpha_k \underline{z} \beta_k$



für $z \in Z$ und $z_e \in E$ und $\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_k \in \Gamma^*$.

(Deterministische) k -Band Turing-Maschine M

Überföhrungsfunktion $\delta: (Z \setminus E) \times \Gamma^k \rightarrow Z \times (\Gamma \times \{L, R, N\})^k$.

Konfiguration $\alpha_1 z \beta_1, \alpha_2 \circ \beta_2, \dots, \alpha_k \circ \beta_k$

Startkonfiguration $\underline{z_0 x_1}, \underline{\circ x_2}, \dots, \underline{\circ x_k}$

für $z \in Z$ und $z_e \in E$ und $\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_k \in \Gamma^*$.

(Deterministische) k -Band Turing-Maschine M

Überföhrungsfunktion $\delta: (Z \setminus E) \times \Gamma^k \rightarrow Z \times (\Gamma \times \{L, R, N\})^k$.

Konfiguration $\alpha_1 z \beta_1, \alpha_2 \circ \beta_2, \dots, \alpha_k \circ \beta_k$

Startkonfiguration $z_0 x_1, \circ x_2, \dots, \circ x_k$

Folgekonfiguration \vdash_M^1 entsprechend...

für $z \in Z$ und $z_e \in E$ und $\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_k \in \Gamma^*$.

(Deterministische) k -Band Turing-Maschine M

Überföhrungsfunktion $\delta: (Z \setminus E) \times \Gamma^k \rightarrow Z \times (\Gamma \times \{L, R, N\})^k$.

Konfiguration $\alpha_1 z \beta_1, \alpha_2 \circ \beta_2, \dots, \alpha_k \circ \beta_k$

Startkonfiguration $z_0 x_1, \circ x_2, \dots, \circ x_k$

Folgekonfiguration \vdash_M^1 entsprechend...

Berechnung von Funktionen $\text{BIN}(x_k)$

$$\underbrace{z_0 x_1}_{\text{BIN}(x_1)}, \underbrace{\circ x_2}_{\text{BIN}(x_2)}, \dots, \underbrace{\circ x_k}_{\text{BIN}(x_k)} \vdash_M^* \underbrace{z_e}_{\text{BIN}(f(x_1, \dots, x_k))}, \underbrace{\alpha_2 \circ \beta_2}_{\text{BIN}(x_2)}, \dots, \underbrace{\alpha_k \circ \beta_k}_{\text{BIN}(x_k)}$$

für $z \in Z$ und $\underbrace{z_e \in E}$ und $\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_k \in \Gamma^*$.

(Deterministische) k -Band Turing-Maschine M

Überföhrungsfunktion $\delta: (Z \setminus E) \times \Gamma^k \rightarrow Z \times (\Gamma \times \{L, R, N\})^k$.

Konfiguration $\alpha_1 z \beta_1, \alpha_2 \circ \beta_2, \dots, \alpha_k \circ \beta_k$

Startkonfiguration $z_0 x_1, \circ x_2, \dots, \circ x_k$

Folgekonfiguration \vdash_M^1 entsprechend...

Berechnung von Funktionen

$$z_0 x_1, \circ x_2, \dots, \circ x_k \vdash_M^* z_e \text{BIN}(f(x_1, \dots, x_k)), \alpha_2 \circ \beta_2, \dots, \alpha_k \circ \beta_k$$

Akzeptanz von Sprachen

$$\underline{z_0 x_1, \circ \square, \dots, \circ \square} \vdash_M^* \alpha_1 z_e \beta_1, \alpha_2 \circ \beta_2, \dots, \alpha_k \circ \beta_k$$

für $z \in Z$ und $\underline{z_e \in E}$ und $\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_k \in \Gamma^*$.

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q simuliert M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:

Band 1 

Band k 

$$(a, a) \in T_Q$$
$$(b, a) \in T_Q$$

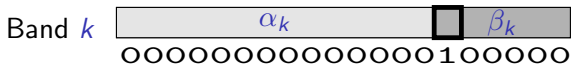
$$T_Q = T_M^k$$

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:

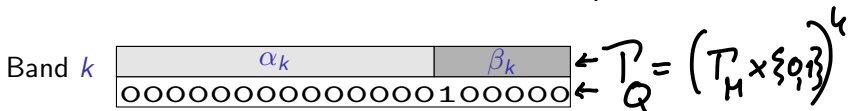
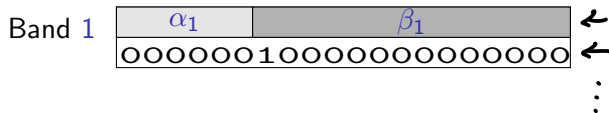


Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:

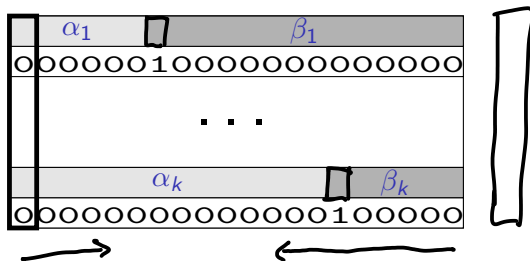


Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

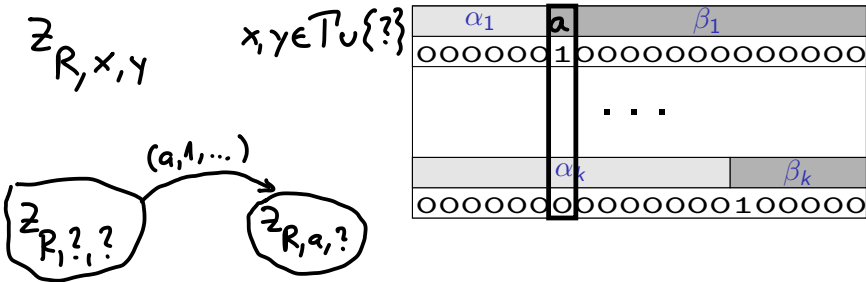
Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



Mehrband-TM Äquivalenz Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



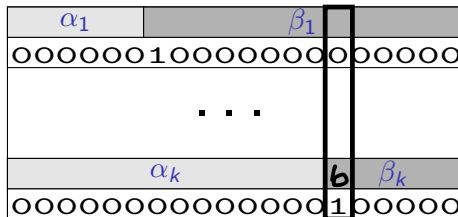
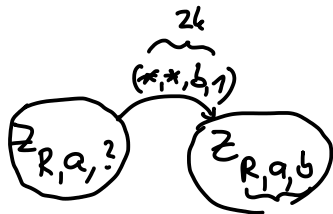
speichere das erste Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



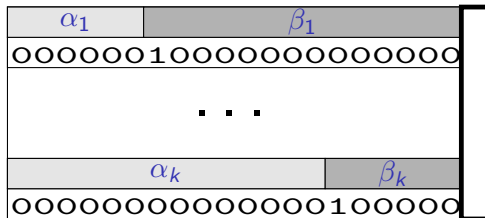
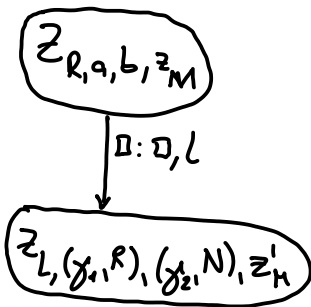
“speichere” das ersten Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



$$\delta: (\underbrace{Z \setminus \Gamma}_{Z_M} \times \underbrace{T_M^k}_{T_M}) \rightarrow Z_M \cup (T_M \times (L, R, N))^k$$

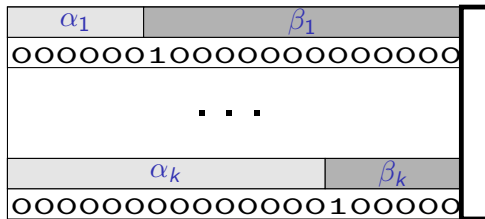
„speichere“ das ersten Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



“speichere” das ersten Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

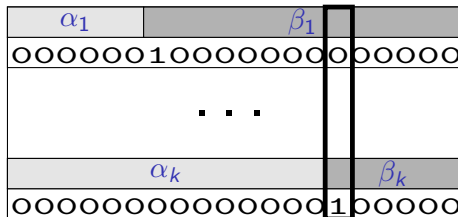
“speichere” neues Zeichen $\gamma_i \in \Gamma$ & Kopfrichtung $d_i \in \{L, R, N\}$ für alle $i \leq k$ im Zustand

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



“speichere” das erste Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

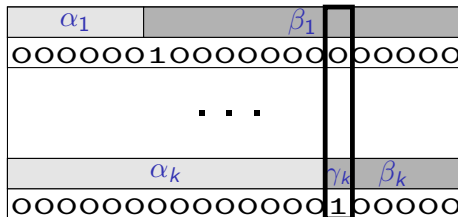
“speichere” neues Zeichen $\gamma_i \in \Gamma$ & Kopfrichtung $d_i \in \{L, R, N\}$ für alle $i \leq k$ im Zustand

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



“speichere” das ersten Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

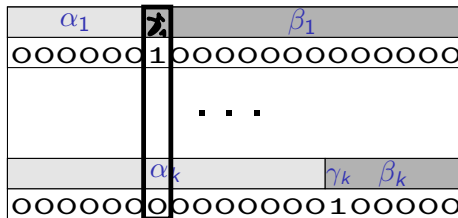
“speichere” neues Zeichen $\gamma_i \in \Gamma$ & Kopfrichtung $d_i \in \{L, R, N\}$ für alle $i \leq k$ im Zustand

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



$z_L, (y_1, R)_{1..}$

“speichere” das ersten Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

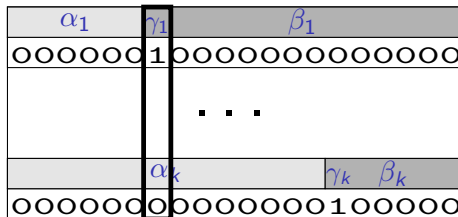
“speichere” neues Zeichen $\gamma_i \in \Gamma$ & Kopfrichtung $d_i \in \{L, R, N\}$ für alle $i \leq k$ im Zustand

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



“speichere” das ersten Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

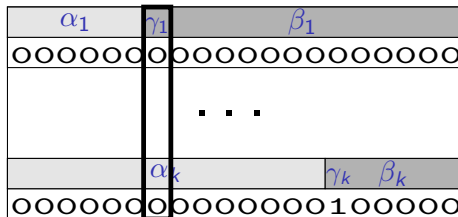
“speichere” neues Zeichen $\gamma_i \in \Gamma$ & Kopfrichtung $d_i \in \{L, R, N\}$ für alle $i \leq k$ im Zustand

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



“speichere” das ersten Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

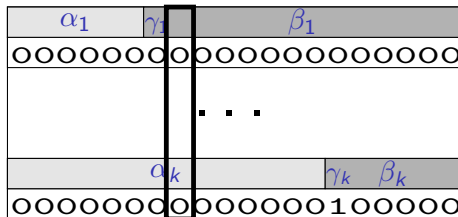
“speichere” neues Zeichen $\gamma_i \in \Gamma$ & Kopfrichtung $d_i \in \{L, R, N\}$ für alle $i \leq k$ im Zustand

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



“speichere” das ersten Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

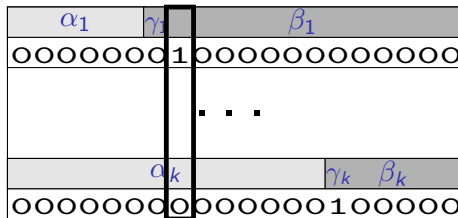
“speichere” neues Zeichen $\gamma_i \in \Gamma$ & Kopfrichtung $d_i \in \{L, R, N\}$ für alle $i \leq k$ im Zustand

Mehrband-TM Äquivalenz

Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



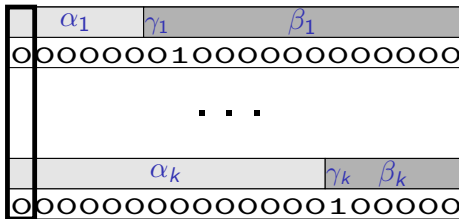
“speichere” das ersten Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

“speichere” neues Zeichen $\gamma_i \in \Gamma$ & Kopfrichtung $d_i \in \{L, R, N\}$ für alle $i \leq k$ im Zustand

Mehrband-TM Äquivalenz Theorem

Zu jeder k -Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Beweisidee: Q **simuliert** M mithilfe eines „fetten Bandes“ mit $2k$ „Spuren“:



“speichere” das erste Zeichen $\beta_i[0] \in \Gamma$ von β_i für alle $i \leq k$ im Zustand

“speichere” neues Zeichen $\gamma_i \in \Gamma$ & Kopfrichtung $d_i \in \{L, R, N\}$ für alle $i \leq k$ im Zustand