Grundbegriffe der Informatik

Tutorium 1 - 13. Sitzung

Dennis Felsing

dennis.felsing@student.kit.edu
http://www.stud.uni-karlsruhe.de/~ubcqr/2010w/tut_gbi/

2011-01-31

Turingmaschinen

Wiederholung

Definition

Eine **Turingmaschine** ist ein Tupel (Z, z_0, X, f, g, m) mit

Wiederholung

Definition

Eine **Turingmaschine** ist ein Tupel (Z, z_0, X, f, g, m) mit

- Zustandsmenge Z
- Anfangszustand $z_0 \in Z$
- Bandalphabet X
- Partielle Zustandsüberführungsfunktion $f : Z \times X \dashrightarrow Z$
- Partielle Ausgabefunktion $g : Z \times X \dashrightarrow X$
- Partielle Bewegungsfunktion $m : Z \times X \dashrightarrow \{-1, 0, 1\}$

Wiederholung

Definition

Eine **Turingmaschine** ist ein Tupel (Z, z_0, X, f, g, m) mit

- Zustandsmenge Z
- Anfangszustand $z_0 \in Z$
- Bandalphabet X
- Partielle Zustandsüberführungsfunktion $f : Z \times X \dashrightarrow Z$
- Partielle Ausgabefunktion $g : Z \times X \dashrightarrow X$
- Partielle Bewegungsfunktion $m : Z \times X \dashrightarrow \{-1, 0, 1\}$

Wie unterscheidet sich eine partielle von einer totalen Funktion?

Wiederholung

Definition

Eine **Turingmaschine** ist ein Tupel (Z, z_0, X, f, g, m) mit

- Zustandsmenge Z
- Anfangszustand $z_0 \in Z$
- Bandalphabet X
- Partielle Zustandsüberführungsfunktion $f : Z \times X \dashrightarrow Z$
- Partielle Ausgabefunktion $g : Z \times X \dashrightarrow X$
- Partielle Bewegungsfunktion $m : Z \times X \dashrightarrow \{-1, 0, 1\}$

Wie unterscheidet sich eine partielle von einer totalen Funktion?
Partielle Funktionen sind nicht linkstotal, müssen also nicht für alle Eingaben definiert sein.

Wiederholung

Definition

Eine **Turingmaschine** ist ein Tupel (Z, z_0, X, f, g, m) mit

- Zustandsmenge Z
- Anfangszustand $z_0 \in Z$
- Bandalphabet X
- Partielle Zustandsüberführungsfunktion $f : Z \times X \dashrightarrow Z$
- Partielle Ausgabefunktion $g : Z \times X \dashrightarrow X$
- Partielle Bewegungsfunktion $m : Z \times X \dashrightarrow \{-1, 0, 1\}$

f,g,m sind dabei für genau die selben Eingabewerte definiert.

Wie unterscheidet sich eine partielle von einer totalen Funktion?
Partielle Funktionen sind nicht linkstotal, müssen also nicht für alle Eingaben definiert sein.

Video

Berechnungskomplexität

Komplexitätsmaße

Definitionen

Konfiguration Gesamtzustand einer TM, Tupel von Zustand, Beschriftung des gesamten Bandes, Kopfposition

$c_0(w)$ Anfangskonfiguration mit Wort w auf Band

$\Delta_t(c)$ Konfiguration der TM nach t Schritten ausgehend von Konfiguration c

$\Delta_*(c)$ Endkonfiguration, falls vorhanden

Definitionen

Wir betrachten im Folgenden nur Turingmaschinen, die für jede Eingabe halten, damit $\Delta_*(c)$ immer definiert ist.

$time_T : A^+ \to \mathbb{N}_+$ mit

$time_T(w) =$ dasjenige t mit $\Delta_t(c_0(w)) = \Delta_*(c_0(w))$

In Worten:

Definitionen

Wir betrachten im Folgenden nur Turingmaschinen, die für jede Eingabe halten, damit $\Delta_*(c)$ immer definiert ist.

$time_T : A^+ \to \mathbb{N}_+$ mit

$time_T(w) =$ dasjenige t mit $\Delta_t(c_0(w)) = \Delta_*(c_0(w))$

In Worten: Wie viele Schritte macht die TM T für eine bestimmte Eingabe w.

Definitionen

Wir betrachten im Folgenden nur Turingmaschinen, die für jede Eingabe halten, damit $\Delta_*(c)$ immer definiert ist.

$time_T : A^+ \to \mathbb{N}_+$ mit

$time_T(w) =$ dasjenige t mit $\Delta_t(c_0(w)) = \Delta_*(c_0(w))$

In Worten: Wie viele Schritte macht die TM T für eine bestimmte Eingabe w.

Zeitkomplexität $Time_T : \mathbb{N}_+ \to \mathbb{N}_+$ mit

$Time_T(n) = max\{time_T(w) \mid w \in A^n\}$

In Worten:

Definitionen

Wir betrachten im Folgenden nur Turingmaschinen, die für jede Eingabe halten, damit $\Delta_*(c)$ immer definiert ist.

$time_T : A^+ \to \mathbb{N}_+$ mit

$time_T(w) =$ dasjenige t mit $\Delta_t(c_0(w)) = \Delta_*(c_0(w))$

In Worten: Wie viele Schritte macht die TM T für eine bestimmte Eingabe w.

Zeitkomplexität $Time_T : \mathbb{N}_+ \to \mathbb{N}_+$ mit

$Time_T(n) = max\{time_T(w) \mid w \in A^n\}$

In Worten: Wie viele Schritte macht die TM T im schlimmsten Fall für Eingaben der Größe n.

Definitionen

Wir betrachten im Folgenden nur Turingmaschinen, die für jede Eingabe halten, damit $\Delta_*(c)$ immer definiert ist.

$time_T : A^+ \to \mathbb{N}_+$ mit

$time_T(w) =$ dasjenige t mit $\Delta_t(c_0(w)) = \Delta_*(c_0(w))$

In Worten: Wie viele Schritte macht die TM T für eine bestimmte Eingabe w.

Zeitkomplexität $Time_T : \mathbb{N}_+ \to \mathbb{N}_+$ mit

$Time_T(n) = max\{time_T(w) \mid w \in A^n\}$

In Worten: Wie viele Schritte macht die TM T im schlimmsten Fall für Eingaben der Größe n.

$space_T(w) : A^+ \to \mathbb{N}_+$ mit $space_T(w) =$ die Anzahl der Felder, die während der Berechnung für Eingabe w benötigt werden

Definitionen

Wir betrachten im Folgenden nur Turingmaschinen, die für jede Eingabe halten, damit $\Delta_*(c)$ immer definiert ist.

$time_T : A^+ \to \mathbb{N}_+$ mit

$time_T(w) =$ dasjenige t mit $\Delta_t(c_0(w)) = \Delta_*(c_0(w))$

In Worten: Wie viele Schritte macht die TM T für eine bestimmte Eingabe w.

Zeitkomplexität $Time_T : \mathbb{N}_+ \to \mathbb{N}_+$ mit

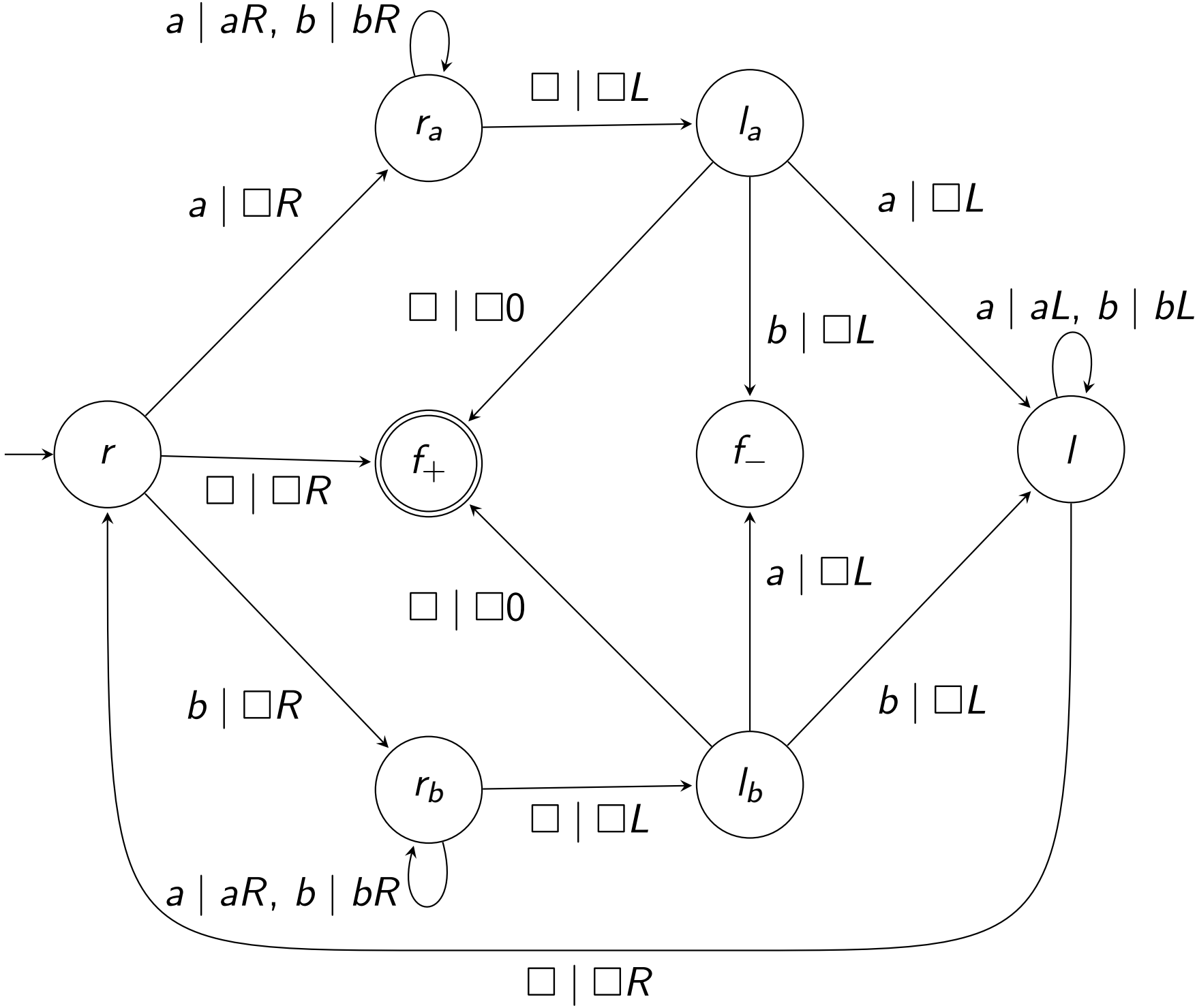
$Time_T(n) = max\{time_T(w) \mid w \in A^n\}$

In Worten: Wie viele Schritte macht die TM T im schlimmsten Fall für Eingaben der Größe n.

$space_T(w) : A^+ \to \mathbb{N}_+$ mit $space_T(w) =$ die Anzahl der Felder, die während der Berechnung für Eingabe w benötigt werden

Platzkomplexität $Space_T(n) : \mathbb{N}_+ \to \mathbb{N}_+$ mit

$Space_T(n) = max\{space_T(w) \mid w \in A^n\}$


$a \mid aR,\ b \mid bR$
r_a
$\square \mid \square L$
l_a
$a \mid \square R$
$a \mid \square L$
$\square \mid \square 0$
$b \mid \square L$
$a \mid aL,\ b \mid bL$
r
f_+
f_-
l
$\square \mid \square R$
$a \mid \square L$
$\square \mid \square 0$
$b \mid \square R$
$b \mid \square L$
r_b
l_b
$\square \mid \square L$
$a \mid aR,\ b \mid bR$
$\square \mid \square R$

Zusammenhänge Platz- und Zeitkomplexität

Wenn eine Turingmaschine für eine Eingabe w $time(w)$ Schritte macht, wie viel Platz kann sie maximal verbrauchen?

Zusammenhänge Platz- und Zeitkomplexität

Wenn eine Turingmaschine für eine Eingabe w $time(w)$ Schritte macht, wie viel Platz kann sie maximal verbrauchen?
$space(w) \leq max(\mid w \mid, 1 + time(w))$

Komplexität einer TM

Sei T_1 eine TM die eine Binärzahl auf dem Band um 1 erhöht und dann wieder an den Anfang des Bandes läuft.
Auf dieser TM basierend entwerfen wir eine neue TM T_2: Auf dem Band steht eine Folge von Nullen. Solange auf dem Band nicht nur Einsen stehen, T_1 anwenden, also die Zahl um 1 erhöhen.

- Welche Platzkomplexität hat T_1 für eine Eingabe der Länge n?
  $Space_{T_1}(n) =$

Komplexität einer TM

Sei T_1 eine TM die eine Binärzahl auf dem Band um 1 erhöht und dann wieder an den Anfang des Bandes läuft.
Auf dieser TM basierend entwerfen wir eine neue TM T_2: Auf dem Band steht eine Folge von Nullen. Solange auf dem Band nicht nur Einsen stehen, T_1 anwenden, also die Zahl um 1 erhöhen.

- Welche Platzkomplexität hat T_1 für eine Eingabe der Länge n?
  $Space_{T_1}(n) = n + 2 \in \Theta(n)$
- $Time_{T_1}(n) =$

Komplexität einer TM

Sei T_1 eine TM die eine Binärzahl auf dem Band um 1 erhöht und dann wieder an den Anfang des Bandes läuft.
Auf dieser TM basierend entwerfen wir eine neue TM T_2: Auf dem Band steht eine Folge von Nullen. Solange auf dem Band nicht nur Einsen stehen, T_1 anwenden, also die Zahl um 1 erhöhen.

- Welche Platzkomplexität hat T_1 für eine Eingabe der Länge n?
  $Space_{T_1}(n) = n + 2 \in \Theta(n)$
- $Time_{T_1}(n) = 2n + 1 \in \Theta(n)$
- $Space_{T_2}(n) =$

Komplexität einer TM

Sei T_1 eine TM die eine Binärzahl auf dem Band um 1 erhöht und dann wieder an den Anfang des Bandes läuft.
Auf dieser TM basierend entwerfen wir eine neue TM T_2: Auf dem Band steht eine Folge von Nullen. Solange auf dem Band nicht nur Einsen stehen, T_1 anwenden, also die Zahl um 1 erhöhen.

- Welche Platzkomplexität hat T_1 für eine Eingabe der Länge n?
  $Space_{T_1}(n) = n + 2 \in \Theta(n)$
- $Time_{T_1}(n) = 2n + 1 \in \Theta(n)$
- $Space_{T_2}(n) = n + 2 \in \Theta(n)$
- $Time_{T_2}(n) \in$

Komplexität einer TM

Sei T_1 eine TM die eine Binärzahl auf dem Band um 1 erhöht und dann wieder an den Anfang des Bandes läuft.
Auf dieser TM basierend entwerfen wir eine neue TM T_2: Auf dem Band steht eine Folge von Nullen. Solange auf dem Band nicht nur Einsen stehen, T_1 anwenden, also die Zahl um 1 erhöhen.

- Welche Platzkomplexität hat T_1 für eine Eingabe der Länge n?
  $Space_{T_1}(n) = n + 2 \in \Theta(n)$
- $Time_{T_1}(n) = 2n + 1 \in \Theta(n)$
- $Space_{T_2}(n) = n + 2 \in \Theta(n)$
- $Time_{T_2}(n) \in \Theta((2^n - 1) \cdot n) = \Theta(2^n \cdot n)$

Komplexitätsklassen

- Bisher haben wir die Zeit- und Platzkomplexität von Turingmaschinen/Algorithmen betrachtet.
- Eine Komplexitätsklasse ist eine Menge von Problemen, die sich mit ähnlichem asymptotischem Aufwand lösen lassen.
- Eine Komplexitätsklasse ist keine Menge von Algorithmen!

Problem vs Algorithmus

Problem

Sortiere eine Folge von n ganzen Zahlen.

Algorithmus

Gehe alle Zahlen durch und füge sie in eine anfangs Reihe ein. Dabei die Reihe von vorne durchgehen bis die richtige Stelle zum Einfügen gefunden wurde.

Aufwand

Problem vs Algorithmus

Problem

Sortiere eine Folge von n ganzen Zahlen.

Algorithmus

Gehe alle Zahlen durch und füge sie in eine anfangs Reihe ein. Dabei die Reihe von vorne durchgehen bis die richtige Stelle zum Einfügen gefunden wurde.

Aufwand

n Operationen mit Aufwand $O(n) \Rightarrow O(n^2)$

Problem vs Algorithmus

Problem

Sortiere eine Folge von n ganzen Zahlen.

Algorithmus

Gehe alle Zahlen durch und füge sie in eine anfangs Reihe ein. Dabei die Reihe von vorne durchgehen bis die richtige Stelle zum Einfügen gefunden wurde.

Aufwand

n Operationen mit Aufwand $O(n) \Rightarrow O(n^2)$

Es gibt aber auch Algorithmen, die das Problem in $O(n \cdot logn)$ lösen können.

Komplexitätsklassen

Definition

P Menge der Entscheidungsprobleme, die sich von Turingmaschinen mit polynomieller Zeitkomplexität lösen lassen

PSPACE Menge der Entscheidungsprobleme, die sich von Turingmaschinen mit polynomieller Platzkomplexität lösen lassen

Aus unserer Erkenntnis $space(w) \leq max(\mid w \mid, 1 + time(w))$ folgt:

$P \subseteq PSPACE$

Unentscheidbare Probleme

Unentscheidbare Probleme

Definition

Ein Problem heißt *unentscheidbar*, wenn es keinen Algorithmus gibt, der es für eine beliebige Eingabe lösen kann, egal wie viel Zeit man ihm lässt.

Postsches Korrespondenzproblem

Gegeben

Eine endliche Folge von Paaren $((x_1, y_1), (x_2, y_2), ..., (x_n, y_n))$ von nicht-leeren Wörtern

Gesucht

Eine nicht-leere Folge $i_1, i_2, ..$ so dass $x_{i_1} \cdot x_{i_2} \cdot ... = y_{i_1} \cdot y_{i_2} \cdot ...$

Beispiel 1

$((1, 101), (10, 00), (011, 11))$

Postsches Korrespondenzproblem

Gegeben

Eine endliche Folge von Paaren $((x_1, y_1), (x_2, y_2), ..., (x_n, y_n))$ von nicht-leeren Wörtern

Gesucht

Eine nicht-leere Folge $i_1, i_2, ..$ so dass $x_{i_1} \cdot x_{i_2} \cdot ... = y_{i_1} \cdot y_{i_2} \cdot ...$

Beispiel 1

$((1, 101), (10, 00), (011, 11))$
Lösung: $1, 3, 2, 3$

Postsches Korrespondenzproblem

Beispiel 2

$((001, 0), (01, 011), (01, 101), (10, 001))$

Postsches Korrespondenzproblem

Beispiel 2

$((001, 0), (01, 011), (01, 101), (10, 001))$

Lösung: 2,4,3,4,4,2,1,2,4,3,4,3,4,4,3,4,4,2,1,4,4,2,1,3,4,1,1,3,4,4,4,
2,1,2,1,1,1,3,4,3,4,1,2,1,4,4,2,1,4,1,1,3,4,1,1,3,1,1,3,1,2,1,4,1,1,3

Postsches Korrespondenzproblem

Beispiel 2

$((001, 0), (01, 011), (01, 101), (10, 001))$
Lösung: 2,4,3,4,4,2,1,2,4,3,4,3,4,4,3,4,4,2,1,4,4,2,1,3,4,1,1,3,4,4,4,
2,1,2,1,1,1,3,4,3,4,1,2,1,4,4,2,1,4,1,1,3,4,1,1,3,1,1,3,1,2,1,4,1,1,3

Die Lösungen können also sehr lang werden. Würde man eine Turingmaschine entwerfen, die das Problem löst, so wüsste man nicht ob sie unendlich lang weiter läuft oder irgendwann anhält.

Fazit

Das Postsche Korrespondenzproblem ist unentscheidbar.

Halteproblem

Eingabe

Eine codierte Turingmaschine und eine Eingabe.

Problem

Hält die codierte TM auf der Eingabe oder läuft sie unendlich weiter?

Behauptung

Das Halteproblem ist unentscheidbar.

Beweis der Unentscheidbarkeit des Halteproblems

Seien $x_0, x_1, x_2, \ldots$ alle möglichen Turingmaschinen $T_{x_0}, T_{x_1}, T_{x_2}, \ldots$ in codierter Form und $f_0, f_1, f_2, \ldots$ die entsprechenden Ausgaben der Turingmaschinen. $f_i(x_j)$ ist undefiniert, wenn die TM für die Eingabe x_j nicht hält.

	x_0	x_1	x_2	x_3	x_4	...
f_0	$f_0(x_0)$	$f_0(x_1)$	$f_0(x_2)$	$f_0(x_3)$	$f_0(x_4)$	$\ldots$
f_1	$f_1(x_0)$	$f_1(x_1)$	$f_1(x_2)$	$f_1(x_3)$	$f_1(x_4)$	$\ldots$
f_2	$f_2(x_0)$	$f_2(x_1)$	$f_2(x_2)$	$f_2(x_3)$	$f_2(x_4)$	$\ldots$
f_3	$f_3(x_0)$	$f_3(x_1)$	$f_3(x_2)$	$f_3(x_3)$	$f_3(x_4)$	$\ldots$
f_4	$f_4(x_0)$	$f_4(x_1)$	$f_4(x_2)$	$f_4(x_3)$	$f_4(x_4)$	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

In dieser Tabelle befinden sich die Ausgaben aller Turingmaschinen für alle möglichen eingegebenen Turingmaschinen.

Beweis der Unentscheidbarkeit des Halteproblems

Wir wollen von nun an mit einem konkreten Beispiel arbeiten:

	x_0	x_1	x_2	x_3	x_4	...
f_0	8	X	3	4	9	...
f_1	7	9	1	X	0	...
f_2	0	5	X	X	X	...
f_3	1	2	X	3	9	...
f_4	6	4	1	9	X	...
⋮	⋮	⋮	⋮	⋮	⋮	⋱

X symbolisiert, dass die Turingmaschine für die Eingabe nicht hält.

Beweis der Unentscheidbarkeit des Halteproblems

Betrachten wir die Diagonale der Tabelle:

	x_0	x_1	x_2	x_3	x_4	...
f_0	8					...
f_1		9				...
f_2			X			...
f_3				3		...
f_4					X	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Definition

$d[i] = f_i(x_i)$, also $d = (8, 9, X, 3, X, ...)$

$$\overline{d}[i] = \overline{f_i(x_i)} = \begin{cases} 1 & \text{falls } d = X \\ X & \text{sonst} \end{cases}, \text{ also } \overline{d} = (X, X, 1, X, 1, ...).$$

Beweis der Unentscheidbarkeit des Halteproblems

Was nützt uns $\overline{d}$?
Jede Zeile in der Tabelle unterscheidet sich von $\overline{d}$. In der Tabelle stehen aber alle von Turingmaschinen berechenbaren Funktionen.
Also gibt es keine Turingmaschine, die $\overline{d}$ berechnet.
Sei T_h die Turingmaschine, die das Halteproblem entscheidet und f_h ihre Ausgabe. Für sie gilt:

$$f_h(f_i(x_i)) = \begin{cases} 1 & \text{falls } T_{x_i} \text{ bei Eingabe } x_i \text{ hält} \\ 0 & \text{sonst} \end{cases}$$

Beweis der Unentscheidbarkeit des Halteproblems

$$f_h(f_i(x_i)) = \begin{cases} 1 & \text{falls } T_{x_i} \text{ bei Eingabe } x_i \text{ hält} \\ 0 & \text{sonst} \end{cases}$$

Wenn man eine solche TM T_h hat, dann kann man eine Turingmaschine $T_{\overline{d}}$ entwerfen, die sich so verhält:

- Für eine Eingabe x_i berechnet $T_{\overline{d}}$ welches Ergebnis T_h für diese Eingabe liefern würde
- Wenn T_h mitteilt, dass $T_{x_i}(x_i)$ hält, dann geht $T_{\overline{d}}$ in eine Endlosschleife
- Wenn T_h mitteilt, dass $T_{x_i}(x_i)$ nicht hält, dann hält $T_{\overline{d}}$ und gibt 1 aus

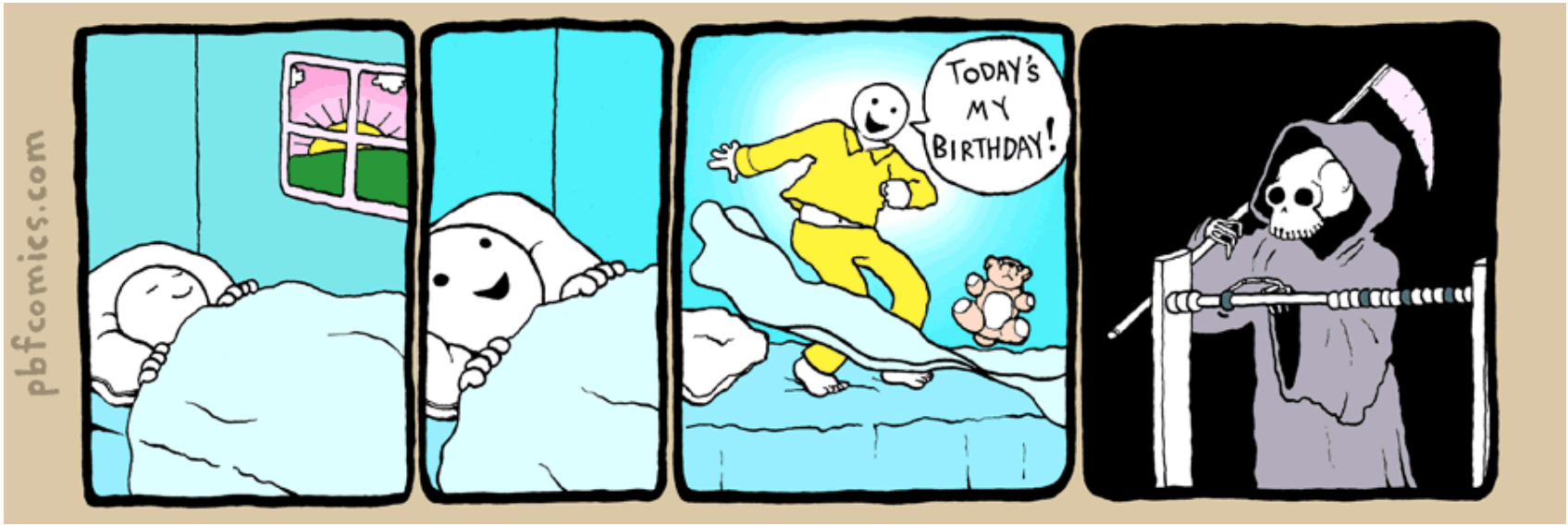
$T_{\overline{d}}$ ist somit genau die zu $\overline{d}$ passende Turingmaschine.

Beweis der Unentscheidbarkeit des Halteproblems

Lediglich unter Verwendung von T_h können wir also eine Turingmaschine entwerfen, die in der Tabelle durch die Zeile $\overline{d}$ repräsentiert werden würde.
Wir wissen aber, dass eine solche Turingmaschine nicht existieren kann, da $\overline{d}$ per Definition von allen Zeilen in der Tabelle verschieden ist.
Also war die Annahme, dass es eine Turingmaschine T_h gibt, die das Halteproblem entscheidet, falsch.
Somit ist das Halteproblem unentscheidbar.

Überblick

1. **Turingmaschinen**
   - Wiederholung
   - Video
2. **Berechnungskomplexität**
   - Komplexitätsmaße
   - Komplexitätsklassen
3. **Unentscheidbare Probleme**
   - Definition
   - Postsches Korrespondenzproblem
   - Halteproblem


pbfcomics.com
TODAY'S MY BIRTHDAY!