Hochschule RheinMain

Automatentheorie und Formale Sprachen

für die Studiengänge
- Angewandte Informatik
- Informatik - Technische Systeme

09 Berechenbarkeit

Prof. Dr. David Sabel
Sommersemester 2025

Stand der Folien: 25. Juni 2025

Hochschule RheinMain

INTUITIVE BERECHENBARKEIT

Intuitive Berechenbarkeit

Hochschule RheinMain

Motivation / Fragen

- Was **kann** mit einem Computerprogramm berechnet werden?
- Was **kann nicht** mit einem Computerprogramm berechnet werden?
- Aus der (Programmier-)Erfahrung:

  Man hat ein gewisses Gefühl dafür, was berechnet werden kann (und was nicht).
- Das ist der intuitive Begriff der Berechenbarkeit.
- Wie beweist man, dass etwas nicht berechnet werden kann?

  Diese Frage ist auch von praktischem Nutzen:
  Suche nach „passendem“ Algorithmus ist sinnlos.

Historie

Hochschule RheinMain

- Berühmte Mathematiker / Informatiker versuchten in den 1930er Jahren den Begriff der Berechenbarkeit zu formalisieren
- dafür entwarfen sie verschiedene Modelle
- insbesondere sind zu nennen:
  - Alan Turing (Turingmaschine)
  - Alonzo Church (Lambda-Kalkül)

Berechenbare Funktion

Hochschule RheinMain

Eine Funktion $f : \mathbb{N}_0^k \to \mathbb{N}_0$ nennen wir **berechenbar**, wenn es einen Algorithmus einer modernen Programmiersprache gibt, der f berechnet:

- Bei Eingabe $(n_1, \ldots, n_k)$ stoppt der Algorithmus nach endlich vielen Berechnungsschritten und liefert den Wert von $f(n_1, \ldots, n_k)$ als Ausgabe.
- Wenn $f(n_1, \ldots, n_k)$ undefiniert ist (f ist also eine partielle Funktion), dann stoppt der Algorithmus nicht.

Genauer: Später, Turing-berechenbar

Beispiele

Hochschule RheinMain

Der Algorithmus

Eingabe: Zahlen $n_1, n_2 \in \mathbb{N}_0$
Beginn
  hilf $:= n_1 + n_2$;
  return *hilf*

berechnet die Funktion $f_1 : (\mathbb{N}_0 \times \mathbb{N}_0) \to \mathbb{N}_0$ mit $f_1(x, y) = x + y$.

Beispiele (2)

Hochschule RheinMain

$$f_2(n) = \begin{cases} 1, & \text{falls es unendlich viele Primzahlzwillinge gibt} \\ 0, & \text{sonst} \end{cases}$$

Ist f_2 berechenbar?

Ein Primzahlzwilling sind zwei Primzahlen deren Differenz 2 ist.

Es ist unbekannt, ob es unendlich viele davon gibt.

Ja: Entweder der Algorithmus, der immer 1 liefert, oder der Algorithmus, der immer 0 liefert, berechnet f_2.

Beispiele (3)

Hochschule RheinMain

$$f^r(n) = \begin{cases} 1, & \text{falls } n \text{ ein Präfix der Ziffern der} \\ & \text{Dezimalzahldarstellung von } r \text{ ist} \\ 0, & \text{sonst} \end{cases}$$

Ist f^r für jedes $r \in \mathbb{R}$ berechenbar?

Nein: Wir bräuchten genauso viele verschiedene Algorithmen wie es reelle Zahlen gibt.
Es gibt nur abzählbar viele Algorithmen einer Programmiersprache, aber überabzählbar viele reelle Zahlen

Hochschule RheinMain

TURINGBERECHENBARKEIT

Turingmaschinen: Wiederholung

Hochschule RheinMain

- Ein Turingmaschine ist ein 7-Tupel $(Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit Zuständen Z, Eingabealphabet Σ, Bandalphabet $\Gamma \supset \Sigma$, Blank-Symbol $\square$, Startzustand z_0, Endzuständen $E \subseteq Z$, und Überführungsfunktion δ.
- DTM: $\delta : (Z \times \Gamma) \to (Z \times \Gamma \times \{L, R, N\})$
- NTM: $\delta : (Z \times \Gamma) \to \mathcal{P}(Z \times \Gamma \times \{L, R, N\})$
- TM-Konfiguration $a_1 \cdots a_m z a_{m+1} \cdots a_n$, wobei $a_1 \cdots a_n \in \Gamma^*$ der entdeckte Teil des Bands, TM ist in Zustand z und Kopf ist unter a_{m+1}.

Turingberechenbarkeit

Hochschule RheinMain

Definition (Turingberechenbarkeit)

Sei $bin(n)$ die Binärdarstellung von $n \in \mathbb{N}_0$.

Eine Funktion $f : \mathbb{N}_0^k \to \mathbb{N}_0$ heißt Turingberechenbar, falls es eine (deterministische) Turingmaschine $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, so dass für alle $n_1, \ldots, n_k, m \in \mathbb{N}_0$ gilt:

$$f(n_1, \ldots, n_k) = m$$
g.d.w.
$$z_0 bin(n_1) \# \ldots \# bin(n_k) \vdash^* \square \ldots \square z_e bin(m) \square \ldots \square \text{ mit } z_e \in E.$$

Eine Funktion $f : \Sigma^* \to \Sigma^*$ heißt Turingberechenbar, falls es eine (deterministische) Turingmaschine $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ gibt, so dass für alle $u, v \in \Sigma^*$ gilt

$$f(u) = v \text{ g.d.w. } z_0 u \vdash^* \square \ldots \square z_e v \square \ldots \square \text{ mit } z_e \in E.$$

Konsequenzen

Hochschule RheinMain

Eine Konsequenz der Definition ist:
Falls $f(n_1, \ldots, n_k)$ bzw. $f(u)$ undefiniert ist,
dann muss die Maschine in eine Endlosschleife gehen.

Churchsche These

Hochschule RheinMain

Verschiedene Berechenbarkeitsmodelle: Turingmaschinen, μ-rekursive Funktionen, Lambda-Kalkül

Resultat: Alle führen zum selben Begriff der Berechenbarkeit.

Churchsche These:

Die Klasse der Turingberechenbaren Funktionen stimmt genau mit der Klasse der intuitiv berechenbaren Funktionen überein.

Die Churchsche These kann man nicht beweisen, da der Begriff „intuitiv berechenbar“ nicht formal gefasst werden kann.

Beispiele

Hochschule RheinMain

Nachfolgerfunktion

Die Funktion $f(x) = x + 1$ für alle $x \in \mathbb{N}_0$ ist Turingberechenbar.
Wir haben eine entsprechende Turingmaschine bereits angegeben.

Überall undefinierte Funktion

Die Funktion $f(x) = \bot$ für alle x ist Turingberechenbar, da die TM $M = (\{z_0\}, \{0, 1, \#\}, \{0, 1, \#, \square\}, \delta, z_0, \square, \emptyset)$ mit $\delta(z_0, a) = (z_0, a, N)$ für keine Eingabe akzeptiert (sondern stets in eine Endlosschleife geht).

Quiz 1

Hochschule RheinMain

Welche Funktion auf Zahlen berechnet die Turing-Maschine

$$(\{z_0\}, \{0, 1, \#\}, \{0, 1, \#, \square\}, \delta, z_0, \square, \{z_0\})?$$

wobei $\delta(z_0, i) = (z_0, i, N)$ für $i \in \{0, 1, \#\}$

arsnova.hs-rm.de
6750 1376

Quiz 2

Hochschule RheinMain

Welche Funktion $f : \Sigma^* \to \Sigma^*$ berechnet die Turingmaschine

$$(\{z_0, z_1\}, \{a\}, \{a, \square\}, \delta, z_0, \square, \{z_1\})$$

mit

- $\delta(z_0, a) = (z_1, \square, R)$
- $\delta(z_0, \square) = (z_1, \square, R)$
- $\delta(z_1, a) = (z_1, a, N)$
- $\delta(z_1, \square) = (z_1, \square, N)$

arsnova.hs-rm.de
6750 1376

Eine nicht-berechenbare Funktion (1)

Hochschule RheinMain

Fleißiger Bieber (busy beaver) mit n Zuständen:

- auf leerer Eingabe anhaltende Turingmaschine
- für die **keine andere** Turingmaschine bei leerer Eingabe **mehr Einsen** auf das Band **schreibt** und **anhält**.

Änderungen / Einschränkungen ggü. unseren TMs:

- Die Turingmaschinen sind deterministisch
- Bandalphabet ist $\{\square, 1\}$, wobei $\square$ das Blank-Symbol ist
- Kopfbewegungen sind nur L und R (N ist nicht erlaubt)
- TM hat n Nichtendzustände und genau einen Endzustand (d.h. in n zählt der Endzustand nicht mit)

Eine nicht-berechenbare Funktion (2)

Hochschule RheinMain

Radó-Funktion (benannt nach Tibor Radó):

$\Sigma(n)$ = Anzahl an Einsen, die ein Fleißiger Bieber mit n Zuständen auf das Band schreibt

Bekannte Werte der Radó-Funktion:

$\Sigma(1)$	$=$	1	Link
$\Sigma(2)$	$=$	4	Link
$\Sigma(3)$	$=$	6	Link
$\Sigma(4)$	$=$	13	Link
$\Sigma(5)$	$=$	4098 nachgewiesen im Jahr 2024	Link
$\Sigma(x)$	$=$	unbekannt, für $x > 5$	
$\Sigma(6)$	$>$	Potenzturm $10^{10^{\cdot^{\cdot^{\cdot^{10}}}}}$ mit 15 Zehnern	

Eine nicht-berechenbare Funktion (3)

Hochschule RheinMain

Satz

Die Radó-Funktion ist nicht berechenbar.

Beweisskizze: Alle TMs im Beweis sind mit den Einschränkungen an fleißige Bieber

- Annahme Σ ist berechenbar.
  Sei $\boldsymbol{M_\Sigma}$ DTM und M_Σ berechnet Σ.
  Sei k_1 die Anzahl an Nichtendzuständen von M_Σ
- TM $\boldsymbol{M_{2x+1}}$ schreibt bei Eingabe 1^x die Ausgabe 1^{2x+1} auf das Band (M_1 verdoppelt alle Einsen und schreibt noch eine zusätzliche Eins).
  Sei k_2 die Anzahl an Nichtendzuständen von M_{2x+1}
- Sei $N = k_1 + k_2$
- TM $\boldsymbol{M_N}$ schreibt 1^N auf das Band.
  Das geht mit N Nichtendzuständen: $\delta(z_i, \square) = (z_{i+1}, 1, R)$, Zustände $z_0, \ldots, z_N$ und z_N ist Endzustand.
  ...

Eine nicht-berechenbare Funktion (4)

Hochschule RheinMain

...

- Sei $\boldsymbol{M}$ die TM, die M_N, dann M_{2x+1} und dann M_Σ ausführt
- M kann mit $N + N = 2N$ Zuständen einfach konstruiert werden.
- Bei leerer Eingabe schreibt M, $\Sigma(2N + 1)$ Einsen auf das Band und hält.
- Widerspruch: M hat nur $2N$ Nichtendzustände, müsste nach Definition von $\Sigma(\cdot)$ mindestens $2N + 1$ Zustände besitzen
- Annahme war falsch: M_Σ existiert nicht, Σ ist nicht berechenbar. $\square$

Hochschule RheinMain

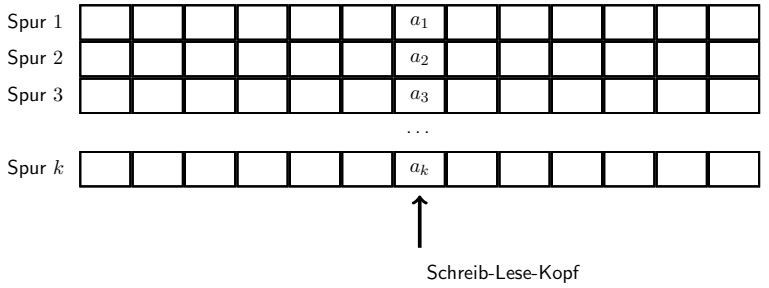
VARIANTEN VON TURINGMASCHINEN

- Mehrspuren-Turingmaschinen
- Mehrband-Turingmaschinen

Mehrspuren-Turingmaschinen (1)

Hochschule RheinMain

Erweiterung: Das Band hat k-Spuren:



Mehrspuren-Turingmaschinen (2)

Hochschule RheinMain

Definition (Mehrspuren-Turingmaschine)

Eine k-**Spuren-Turingmaschine** ($k \in \mathbb{N}$) ist ein 7-Tupel $(Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit

- Z ist eine endliche Menge von Zuständen,
- Σ ist das (endliche) Eingabealphabet,
- $\Gamma \supset \Sigma$ ist das (endliche) Bandalphabet,
- δ ist die Zustandsüberführungsfunktion:
  - für eine DTM: $\delta : (Z \times \Gamma^k) \to Z \times \Gamma^k \times \{L, R, N\}$
  - für eine NTM: $\delta : (Z \times \Gamma^k) \to \mathcal{P}(Z \times \Gamma^k \times \{L, R, N\})$
- $z_0 \in Z$ ist der Startzustand,
- $\square \in \Gamma \setminus \Sigma$ ist das Blank-Symbol und
- $E \subseteq Z$ ist die Menge der Endzustände.

Berechenbarkeit: Ein- und Ausgabe auf der ersten Spur.

Mehrspuren-Turingmaschinen (3)

Hochschule RheinMain

Berechenbarkeit: Ein- und Ausgabe auf der ersten Spur.

Satz

Jede Mehrspuren-Turingmaschine kann auf einer 1-Band-Turingmaschine simuliert werden.

Beweis: Konstruktion der 1-Band-TM mit einer Spur:

- Verwende als Bandalphabet $\Gamma \cup \Gamma^k$
- Eingabealphabet Σ und Blank-Symbol $\square$.
- Aus Eingabe w aus Σ^* erzeugt die TM die Mehrspurendarstellung: Ersetze $a \in \Sigma$ durch k-Tupel $(a, \square, \ldots, \square)$.
- Anschließend wird die Mehrspurenmaschine simuliert
- Nach Akzeptanz der Mehrspurenmaschine: Erzeuge 1-Spuren-Darstellung, d.h. ersetze alle $(a_1, \ldots, a_k)$ durch a_1. □

Beispiel einer Mehrspurenmaschine (1)

Hochschule RheinMain

- TM, die $\{wcw \mid w \in \{a,b\}^+\}$ erkennt.
- Wir konzentrieren uns auf die 2-Spurendarstellung, daher:
  TM, die $\{w(c,\square)w \mid w \in \{(a,\square),(b,\square)\}^+\}$ erkennt

Ideen:

- Eingabe wcw auf Spur 1 wird nur gelesen nicht verändert.
- Auf Spur 2 wird nur $\square$ und ✓ zum Markieren von Zeichen auf Spur 1 verwendet.
- Markieren: Erst ein Zeichen im linken w, dann selbes Zeichen im rechten w
- Zustände $Z = \{z_0, z_{1a}, z_{1b}, z_{2a}, z_{2b}, z_3, \ldots, z_9\}$
  $z_{1a}, z_{1b}, z_{2a}, z_{2b}$ „speichern“ das gelesene Zeichen (a oder b).
  - z_0 ist der Startzustand.
  - z_8 ist der einzige akzeptierende Zustand.
  - z_9 ist Müllzustand (zum Verwerfen)

Beispiel einer Mehrspurenmaschine (2)

Hochschule RheinMain

Übergangsfunktion δ

- $\delta(z_0,(s,\square)) = (z_{1s},(s,✓),R)$ für $s \in \{a,b\}$: markiere linkestes unmarkiertes Zeichen s im linken w, speichere s in z_{1s}, Suche nach s im rechten w beginnt.
- $\delta(z_{1s},(s',\square)) = (z_{1s},(s',\square),R)$ für $s,s' \in \{a,b\}$: laufe nach rechts zum c
- $\delta(z_{1s},(c,\square)) = (z_{2s},(c,\square),R)$ mit $s \in \{a,b\}$: c wurde erreicht
- $\delta(z_{2s},(s',✓)) = (z_{2s},(s',✓),R)$ mit $s,s' \in \{a,b\}$: suche erstes unmarkiertes Symbol im rechten w.
- $\delta(z_{2s},(s,\square)) = (z_3,(s,✓),L)$ mit $s \in \{a,b\}$: richtiges unmarkiertes Zeichen im rechten w .
- $\delta(z_3,(s,✓)) = (z_3,(s,✓),L)$ mit $s \in \{a,b\}$: nach links laufen zum c.
- $\delta(z_3,(c,\square)) = (z_4,(c,\square,L))$ c von rechts wieder erreicht.
- ...

Beispiel einer Mehrspurenmaschine (3)

Hochschule RheinMain

- ...
- $\delta(z_4,(s,\square)) = (z_5,(s,\square),L)$ mit $s \in \{a,b\}$: links vom c kein markiertes Zeichen: zu z_5
- $\delta(z_4,(s,✓)) = (z_6,(s,\square),R)$ mit $s \in \{a,b\}$: links vom c ist markiertes Zeichen: Prüfe, dass alle Zeichen im rechten w markiert sind (mit z_6).
- $\delta(z_5,(s,\square)) = (z_5,(s,\square),L)$ mit $s \in \{a,b\}$: suche erstes markiertes Zeichen im linken w
- $\delta(z_5,(s,✓)) = (z_0,(s,✓),R)$ mit $s \in \{a,b\}$: erstes markiertes Zeichen im linken w gefunden , gehe zu z_0
- $\delta(z_6,(c,\square)) = (z_7,(c,\square),R)$: suche im rechten w nach unmarkierten Zeichen
- $\delta(z_7,(s,✓)) = (z_7,(s,✓),R)$ für $s \in \{a,b\}$: suche im rechten w nach unmarkierten Zeichen
- $\delta(z_7,(\square,\square)) = (z_8,(\square,\square),N)$: alle Zeichen im rechten w markiert, akzeptiere in z_8
- $\delta(z_8,(s,t)) = (z_8,(s,t),N)$ für $s \in \{a,b,c\}$, $t \in \{\square,✓\}$ verbleibe akzeptierend.
- $\delta(z_i,(s,t)) = (z_9,(s,t),N)$ für alle anderen Fälle verbleibe oder wechsle in den verwerfenden Zustand.

Beispiel einer Mehrspurenmaschine (4)

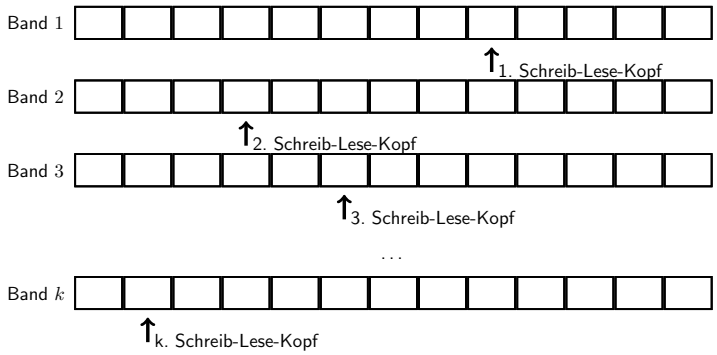
Hochschule RheinMain

Ein Lauf der Turingmaschine für das Wort $abcab$ ist:

$$
\begin{array}{l}
z_0\ \begin{smallmatrix}a&b&c&a&b\\ \square&\square&\square&\square&\square\end{smallmatrix} \vdash \begin{smallmatrix}a\\ ✓\end{smallmatrix} z_{1a}\ \begin{smallmatrix}b&c&a&b\\ \square&\square&\square&\square\end{smallmatrix} \vdash \begin{smallmatrix}a&b\\ ✓&\square\end{smallmatrix} z_{1a}\ \begin{smallmatrix}c&a&b\\ \square&\square&\square\end{smallmatrix} \vdash \begin{smallmatrix}a&b&c\\ ✓&\square&\square\end{smallmatrix} z_{2a}\ \begin{smallmatrix}a&b\\ \square&\square\end{smallmatrix} \\
\vdash \begin{smallmatrix}a&b\\ ✓&\square\end{smallmatrix} z_3\ \begin{smallmatrix}c&a&b\\ \square&✓&\square\end{smallmatrix} \vdash \begin{smallmatrix}a\\ ✓\end{smallmatrix} z_4\ \begin{smallmatrix}b&c&a&b\\ \square&\square&✓&\square\end{smallmatrix} \vdash z_5\ \begin{smallmatrix}a&b&c&a&b\\ ✓&\square&\square&✓&\square\end{smallmatrix} \vdash \begin{smallmatrix}a\\ ✓\end{smallmatrix} z_0\ \begin{smallmatrix}b&c&a&b\\ \square&\square&✓&\square\end{smallmatrix} \\
\vdash \begin{smallmatrix}a&b\\ ✓&✓\end{smallmatrix} z_{1b}\ \begin{smallmatrix}c&a&b\\ \square&✓&\square\end{smallmatrix} \vdash \begin{smallmatrix}a&b&c\\ ✓&✓&\square\end{smallmatrix} z_{2b}\ \begin{smallmatrix}a&b\\ ✓&\square\end{smallmatrix} \vdash \begin{smallmatrix}a&b&c&a\\ ✓&✓&\square&✓\end{smallmatrix} z_{2b}\ \begin{smallmatrix}b\\ \square\end{smallmatrix} \vdash \begin{smallmatrix}a&b&c\\ ✓&✓&\square\end{smallmatrix} z_3\ \begin{smallmatrix}a&b\\ ✓&✓\end{smallmatrix} \\
\vdash \begin{smallmatrix}a&b\\ ✓&✓\end{smallmatrix} z_3\ \begin{smallmatrix}c&a&b\\ \square&✓&✓\end{smallmatrix} \vdash \begin{smallmatrix}a\\ ✓\end{smallmatrix} z_4\ \begin{smallmatrix}b&c&a&b\\ ✓&\square&✓&✓\end{smallmatrix} \vdash \begin{smallmatrix}a&b\\ ✓&✓\end{smallmatrix} z_6\ \begin{smallmatrix}c&a&b\\ \square&✓&✓\end{smallmatrix} \vdash \begin{smallmatrix}a&b&c\\ ✓&✓&\square\end{smallmatrix} z_7\ \begin{smallmatrix}a&b\\ ✓&✓\end{smallmatrix} \\
\vdash \begin{smallmatrix}a&b&c&a\\ ✓&✓&\square&✓\end{smallmatrix} z_7\ \begin{smallmatrix}b\\ ✓\end{smallmatrix} \vdash \begin{smallmatrix}a&b&c&a&b\\ ✓&✓&\square&✓&✓\end{smallmatrix} z_7\ \begin{smallmatrix}\square\\ \square\end{smallmatrix} \vdash \begin{smallmatrix}a&b&c&a&b\\ ✓&✓&\square&✓&✓\end{smallmatrix} z_8\ \begin{smallmatrix}\square\\ \square\end{smallmatrix}
\end{array}
$$

Mehrbandmaschinen

Hochschule RheinMain



Schreib-Lese-Köpfe bewegen sich unabhängig!

Mehrbandmaschinen (2)

Hochschule RheinMain

Definition (Mehrband-Turingmaschine)

Eine k-Band-Turingmaschine (für $k \in \mathbb{N}$) ist ein 7-Tupel $(Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit

- Z ist eine endliche Menge von Zuständen,
- Σ ist das (endliche) Eingabealphabet,
- $\Gamma \supset \Sigma$ ist das (endliche) Bandalphabet,
- δ ist die Zustandsüberführungsfunktion
  - für eine DTM: $\delta : (Z \times \Gamma^k) \to (Z \times \Gamma^k \times \{L, R, N\}^k)$
  - für eine NTM: $\delta : (Z \times \Gamma^k) \to \mathcal{P}(Z \times \Gamma^k \times \{L, R, N\}^k)$
- $z_0 \in Z$ ist der Startzustand,
- $\square \in \Gamma \setminus \Sigma$ ist das Blank-Symbol
- $E \subseteq Z$ ist die Menge der Endzustände.

Berechnung: Eingabe auf dem Band 1, alle anderen leer, Ausgabe auf Band 1

Mehrbandmaschinen (3)

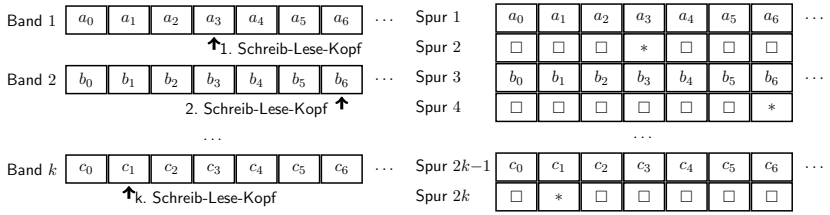
Hochschule RheinMain

Theorem

Jede Mehrband-TM kann von einer 1-Band TM simuliert werden.

Beweis:

- Sei M eine k-Band-TM. Wenn $k = 1$, dann nichts zu zeigen.
- Für $k > 1$ verwenden wir eine $2 \cdot k$-Spuren-TM um M zu simulieren.



k-Band-TM Simulation mit $2k$-Spuren

Mehrbandmaschinen (4)

Hochschule RheinMain

- Eingabe $w \in \Sigma^*$ auf dem Eingabeband der Mehrband-Maschine
- 1-Band-Maschine erzeugt Darstellung in $2k$-Spuren
- 1-Band-Maschine simuliert anschließend Berechnungsschritte.
- Bei Akzeptanz der Mehrband-TM transformiert die 1-Band-TM die Spurendarstellung in die Darstellung der Ausgabe.
- Simulation eines Berechnungsschrittes der Mehrband-TM:
  1) lese den verwendeten Bandbereich von links nach rechts
  2) speichere alle k Kopfpositionen (durch Zustände)
  3) führe Schritt der Mehrband-TM aus, durch Anpassen der Bandinhalte und der Kopfpositionen □

Hochschule RheinMain

ENTSCHEIDBARKEIT

Entscheidbarkeit

Hochschule RheinMain

Definition

Eine Sprache $L \subseteq \Sigma^*$ heißt **entscheidbar**, wenn die charakteristische Funktion von L, $\chi_L : \Sigma^* \to \{0,1\}$ mit

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

berechenbar ist.

algorithmisch:

Der χ_L-berechnende Algorithmus terminiert in jedem Fall und liefert ein Ergebnis.

Semi-Entscheidbarkeit

Hochschule RheinMain

Definition

Eine Sprache heißt **semi-entscheidbar** falls $\chi'_L : \Sigma^* \to \{0,1\}$ mit

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

berechenbar ist.

algorithmisch:

Der χ'_L-berechnende Algorithmus terminiert nur, falls $w \in L$, und läuft anderenfalls endlos.

Zusammenhang: entscheidbar und semi-entscheidbar

Hochschule RheinMain

Satz

Ein Sprache L ist genau dann entscheidbar, wenn L und $\overline{L}$ jeweils semi-entscheidbar sind.

Beweis:

„⇒“: Konstruiere aus TM, die χ_L berechnet, zwei TMs, die χ'_L und $\chi'_{\overline{L}}$ berechnen.

„⇐“: Gegeben TMs M_L und $M_{\overline{L}}$, die χ'_L und $\chi'_{\overline{L}}$ berechnen.
Konstruiere TM, die χ_L berechnet:

- Starte mit $i = 1$.
- Simuliere i-Schritte von M_L.
- Wenn diese akzeptiert, dann akzeptiere mit Ausgabe 1.
- Ansonsten simuliere i-Schritte von $M_{\overline{L}}$.
- Wenn diese akzeptiert, dann akzeptiere mit Ausgabe 0.
- Ansonsten erhöhe i um 1 und starte von neuem.

Komplement entscheidbar?

Hochschule RheinMain

Korollar

Wenn L entscheidbar, dann ist auch $\overline{L}$ entscheidbar.

- da L entscheidbar, sind L und $\overline{L}$ semi-entscheidbar
- daher sind $\overline{\overline{L}} = L$ und $\overline{L}$ semi-entscheidbar
- daher ist $\overline{L}$ entscheidbar

Quiz 3 zur Wiederholung

Hochschule RheinMain

Sei $f : \Sigma^* \to \Sigma^*$. Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine DTM.
M berechnet f wenn ....

- $z_0 w \vdash^* z_e v$ und $z_e \in E$ gdw. $f(w) = v$
- für alle $z_e \in E$: $z_0 w \vdash^* z_e v$ gdw. $f(w) = v$
- $z_0 w \vdash^* z_e v$ und $z_e \in E$ gdw. $f(v) = w$

arsnova.hs-rm.de
6750 1376

Quiz 4 zur Wiederholung

Hochschule RheinMain

Sei $f(x) = 1$, wenn $x \in L$ und $f(x) =$ undefiniert, sonst. f sei Turing-berechenbar. Dann ist...

- L entscheidbar
- Das Komplement $\overline{L}$ semi-entscheidbar
- L semi-entscheidbar

arsnova.hs-rm.de
6750 1376

Quiz 5 zur Wiederholung

Hochschule RheinMain

Sei $f(x) = 0$, wenn $x \in L$ und $f(x) = 1$ wenn $x \notin L$. f sei Turing-berechenbar. Dann ist...

- L semi-entscheidbar
- $\overline{L}$ semi-entscheidbar
- L entscheidbar
- $\overline{L}$ entscheidbar
- Nichts von alledem

arsnova.hs-rm.de
6750 1376

Quiz 6 zur Wiederholung

Hochschule RheinMain

Sei $f(x) = 1$, wenn $x \in L$ und $f(x) = 1$ wenn $x \notin L$. f sei Turing-berechenbar. Dann gilt auf jeden Fall

- L semi-entscheidbar
- $\overline{L}$ semi-entscheidbar
- L entscheidbar
- $\overline{L}$ entscheidbar
- Nichts von alledem

arsnova.hs-rm.de
6750 1376

Rekursive Aufzählbarkeit

Hochschule RheinMain

Definition

Eine Sprache $L \subseteq \Sigma^*$ heißt **rekursiv aufzählbar**,

- falls $L = \emptyset$ oder
- falls es eine totale berechenbare Funktion $f : \mathbb{N}_0 \to \Sigma^*$ gibt, sodass $L = \bigcup_{i \in \mathbb{N}_0} f(i)$.
  Man sagt dann „f zählt L auf“.

Beispiel

Hochschule RheinMain

Lemma

Die Sprache Σ^* ist rekursiv-aufzählbar.

Beweis: Sei $|\Sigma| = b$, $n \in \mathbb{N}_0$. Interpretiere $w \in \Sigma^*$ als $b+1$-äre Zahl.

- Konstruiere TM, die n in Binärdarstellung auf Eingabeband erhält.
- TM erzeugt auf anderem Band die $b+1$-äre Darstellung der 0
- Anschließend zählt die TM die Zahl auf dem Eingabeband um 1 herunter und die $b+1$-äre Zahl um 1 nach oben.
- Dies wird wiederholt bis auf dem Eingabeband die 0 steht
- Dann steht auf dem anderen Band $f(n)$.

Rekursiv aufzählbar = semi-entscheidbar

Hochschule RheinMain

Satz

Eine Sprache ist genau dann rekursiv aufzählbar, wenn sie semi-entscheidbar ist.

Beweis:

„$\Rightarrow$“: Sei f die totale, berechenbare Funktion, die L aufzählt.
Dann berechnet der folgende Algorithmus $\chi'_L(w)$:

Für $i = 0, 1, 2, 3, \ldots$ **tue**
  wenn $f(i) = w$ **dann**
    stoppe und gebe 1 aus

„$\Leftarrow$“: ...

Rekursiv aufzählbar = semi-entscheidbar

„⇐“: Sei M eine TM, die χ'_L berechnet.

- Wenn $L = \emptyset$, dann ist L rekursiv-aufzählbar. Anderenfalls sei $u \in L$.
  Wir konstruieren TM M', die die L aufzählende Funktion berechnet.
- Sei n eine Eingabe. Wir interpretieren n als kodiertes Paar $c(x, y)$
  mit $left(n) = x$ und $right(n) = y$ einer Bijektion $c : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$
- M' simuliert y Schritte von M bei Eingabe $g(x)$, wobei g die Σ^* aufzählende Funktion.
- Wenn M nach y Schritten $g(x)$ akzeptiert, dann akzeptiert M'mit Ausgabe $g(x)$.
  Anderenfalls, akzeptiert M' mit Ausgabe u.
- Die von M' berechnete Funktion:

$$f(n) = \begin{cases} w \in L, \text{falls } w = g(left(n)) \text{ und } M \text{ akzeptiert } w \text{ in } right(n) \text{ Schritten} \\ u \in L, \text{ sonst} \end{cases}$$

- f zählt L auf, da für jedes Wort w ein x existiert mit $g(x) = w$ und
  ein y existiert, sodass M mit Eingabe w nach y Schritten akzeptiert.

Zusammenfassung: Äquivalente Eigenschaften

Die folgenden Eigenschaften sind äquivalent:

- L ist vom Typ 0.
- L ist semi-entscheidbar.
- L ist rekursiv-aufzählbar.
- Es gibt eine Turingmaschine M, die L akzeptiert (d.h. $L(M) = L$).
- χ'_L ist Turing-berechenbar.
- Es gibt berechenbare Funktionen, die L als Wertebereich
  (nämlich die L aufzählende Funktion)
  bzw. als Definitionsbereich (nämlich χ'_L) haben.

Rekursiv aufzählbar ≠ abzählbar

- Sprache L ist abzählbar, wenn es eine totale Funktion $f : \mathbb{N}_0 \to L$ gibt,
  sodass $\bigcup_{i \in \mathbb{N}_0} f(i) = L$.
- Beachte: Abzählbarkeit fordert nicht, dass f berechenbar ist!

Gödelisierung von Turingmaschinen

Ziel:
Stelle Turingmaschinenbeschreibung als
natürliche Zahl in Binärdarstellung dar:

Grund:
Andere Turingmaschinen können die Beschreibung als
Eingabe erhalten, erzeugen, usw.

Gödelisierung von Turingmaschinen (2)

Hochschule RheinMain

Sei $(Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine DTM mit $\Sigma = \{0, 1\}$ und

- $\Gamma = \{a_0, \ldots, a_k\}$ wobei $a_0 = \square$, $a_1 = \#$, $a_2 = 0$, $a_3 = 1$
- $Z = \{z_0, \ldots, z_n\}$
- $E = \{z_n\}$

Für $\delta(z_p, a_i) = (z_q, a_j, D)$ erzeuge Wort über Alphabet $\{0, 1, \#\}$:

$$w_{p,i,q,j,D} = \#\#bin(p)\#bin(i)\#bin(q)\#bin(j)\#bin(D_m)$$

mit $D_m = 0$, falls $D = L$, $D_m = 1$, falls $D = R$, $D_m = 2$, falls $D = N$

Kodierung w_δ: Schreibe alle δ-Wörter hintereinander.

Schließlich: Kodiere Alphabet $\{0, 1, \#\}$ durch $\{0 \mapsto 00, 1 \mapsto 01, \# \mapsto 11\}$.

Wende dies auf w_δ an.

Wir bezeichnen mit w_M die so kodierte TM M.

Gödelisierung von Turingmaschinen (3)

Hochschule RheinMain

- Nicht jedes Wort über $\{0, 1\}$ entspricht der Kodierung einer Turingmaschine.
- Sei $\widehat{M}$ eine beliebige aber feste Turingmaschine.
- Definiere für jedes $w \in \{0, 1\}^*$ die zugehörige TM M_w:

$$M_w := \begin{cases} M, & \text{wenn } w = w_M \\ \widehat{M}, & \text{sonst} \end{cases}$$

Hochschule RheinMain

SPEZIELLES HALTEPROBLEM

- Spezielles Halteproblem
- Unentscheidbarkeit des speziellen Halteproblems

Spezielles Halteproblem

Hochschule RheinMain

Definition (Spezielles Halteproblem)

Das spezielle Halteproblem ist die Sprache

$$K := \{w \in \{0, 1\}^* \mid M_w \text{ hält für Eingabe } w\}$$

Unentscheidbarkeit von K

Hochschule RheinMain

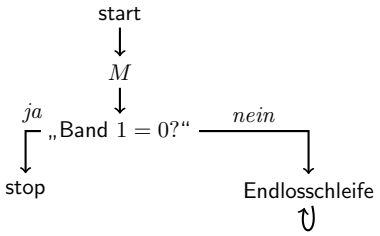
Satz

Das spezielle Halteproblem ist nicht entscheidbar (und damit *unentscheidbar*).

Beweis:

- Annahme: K ist entscheidbar.
- Dann ist χ_K berechenbar, und es gibt TM M, die χ_K berechnet.
- Konstruiere M':

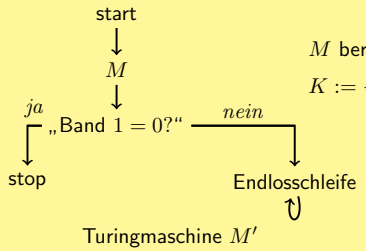
1. M' lässt M ablaufen
2. Wenn M mit 0 auf dem Band endet, dann akzeptiert M'
3. Wenn M mit 1 auf dem Band endet, dann läuft M' in eine Endlosschleife.



Unentscheidbarkeit von K (2)

Hochschule RheinMain

Zur Erinnerung:



M berechnet χ_K wobei K:

$K := \{w \in \{0,1\}^* \mid M_w \text{ hält für Eingabe } w\}$

Betrachte nun M' auf der Eingabe $w_{M'}$:

Es gilt:

	M' hält für Eingabe $w_{M'}$
g.d.w.	M angesetzt auf $w_{M'}$ gibt 0 aus
g.d.w.	$\chi_K(w_{M'}) = 0$
g.d.w.	$w_{M'} \notin K$
g.d.w.	M' hält nicht für Eingabe $w_{M'}$

Unentscheidbarkeit von K (3)

Hochschule RheinMain

- M' hält für Eingabe $w_{M'} \iff M'$ hält nicht für Eingabe $w_{M'}$
- Widerspruch!
- Annahme war falsch!
- K ist nicht entscheidbar, sondern unentscheidbar.

Spezielles Halteproblem

Hochschule RheinMain

Erinnerung:

Das spezielle Halteproblem ist die Sprache

$$K := \{w \in \{0,1\}^* \mid M_w \text{ hält für Eingabe } w\}$$

Beachte: K ist semi-entscheidbar:

Wiederhole für $i = 0, 1, \ldots$:
Simuliere i Schritte von M_w auf Eingabe w
Wenn M_w akzeptiert, dann akzeptiere und gebe 1 aus

Beachte: K unentscheidbar, aber semi-entscheidbar $\implies \overline{K}$ nicht semi-entscheidbar

Der Beweis, dass K unentscheidbar ist, verwendet ein Diagonalisierungsargument, dass wir gleich genauer erläutern.

Diagonalisierungsargument

Hochschule RheinMain

	w_1	w_2	w_3	w_4	$\dots$	w_D
M_{w_1}	ja	$nein$	ja	$\dots$	$\dots$	$\dots$
M_{w_2}	$nein$	$nein$	ja	$\dots$	$\dots$	$\dots$
M_{w_3}	ja	$nein$	ja	$\dots$	$\dots$	$\dots$
M_{w_4}	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
M_{w_D}	$nein$	ja	$nein$	$\dots$	$\dots$	??

- Spalten: alle Wörter über $\{0,1\}$: $w_1, w_2, \dots$
- Zeilen: alle TMs $M_{w_1}, M_{w_2}, \dots$.
- Eintrag in Zeile i und Spalte j: ja, wenn M_{w_i} bei Eingabe w_j akzeptiert, $nein$ sonst.
- Diagonalsprache: $L_D = \{w_i \mid M_{w_i} \text{ hält nicht für Eingabe } w_i\} (= \overline{K})$
- Sei w_D die TM-Beschreibung von TM M_{w_D}, die χ_{L_D} berechnet.
- Eintrag in Zeile M_{w_D} und Spalte w_D ist ja, g.d.w. der Eintrag in Spalte w_D und Zeile M_{w_D} $nein$ ist. Widerspruch! L_D ist daher nicht (semi-)entscheidbar.

Hochschule RheinMain

REDUKTIONEN

- Reduktion
- Unentscheidbarkeit des Halteproblems

Reduktion

Hochschule RheinMain

- Hilfsmittel, um Unentscheidbarkeit nachzuweisen
- Statt Unentscheidbarkeit von Sprache L von Grundauf neu zu beweisen, zeige:
  Wenn man L entscheiden könnte, dann könnte man auch L_0 entscheiden.
- Wenn L_0 bereits als unentscheidbar gezeigt, folgt L ist unentscheidbar.

Reduktion (Definition)

Hochschule RheinMain

Definition (Reduktion (einer Sprache auf eine andere))

Sei $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ Sprachen. Dann sagen wir L_1 ist auf L_2 reduzierbar (geschrieben $L_1 \leq L_2$), falls es eine totale und berechenbare Funktion $f : \Sigma_1^* \to \Sigma_2^*$ gibt, sodass für alle $w \in \Sigma_1^*$ gilt: $w \in L_1 \iff f(w) \in L_2$

Satz

Wenn $L_1 \leq L_2$ und L_2 entscheidbar (bzw. semi-entscheidbar) ist, dann ist auch L_1 entscheidbar (bzw. semi-entscheidbar).

Beweis (nur entscheidbar, semi-entscheidbar analog): Sei f die $L_1 \leq L_2$ bezeugende (und berechenbare) Funktion. Da L_2 entscheidbar, ist χ_{L_2} berechenbar. Damit ist auch $\chi_{L_1}(w) := \chi_{L_2}(f(w))$ berechenbar. Offensichtlich gilt

$$\chi(L_1)(w) := \left\{ \begin{array}{ll} 1, & w \in L_1 \\ 0, & w \notin L_1 \end{array} \right\} = \left\{ \begin{array}{ll} 1, & f(w) \in L_2 \\ 0, & f(w) \notin L_2 \end{array} \right\} = \chi_{L_2}(f(w))$$

Unentscheidbarkeit

Hochschule RheinMain

Mit Kontraposition folgt:

Lemma

Sei $L_1 \leq L_2$ und L_1 ist unentscheidbar.
Dann ist auch L_2 unentscheidbar.

Das ist die Richtung, die wir meistens brauchen:

- L_1 sei eine bekannt unentscheidbare Sprache
- Reduziere L_1 auf L_2 durch Angabe einer berechenbaren Funktion f mit $w \in L_1 \iff f(w) \in L_2$.
- Damit folgt, dass L_2 unentscheidbar ist.

Halteproblem

Hochschule RheinMain

Definition (Halteproblem)

Das (allgemeine) Halteproblem ist $H := \{w\#x \mid \text{TM } M_w \text{ hält für Eingabe } x\}$

Satz

Das allgemeine Halteproblem ist unentscheidbar.

Beweis: Wir reduzieren das spezielle Halteproblem auf das allgemeine Halteproblem, und zeigen daher $K \leq H$. Sei $f(w) = w\#w$. Dann gilt

$$\begin{array}{ll} & w \in K \\ \text{g.d.w.} & M_w \text{ hält für Eingabe } w \\ \text{g.d.w.} & w\#w \in H \\ \text{g.d.w.} & f(w) \in H \end{array}$$

f kann durch eine TM berechnet werden. Damit gilt $K \leq H$ und damit folgt aus K unentscheidbar auch H unentscheidbar.

Halteproblem bei leerer Eingabe

Hochschule RheinMain

Definition

Das Halteproblem auf leerem Band ist $H_0 = \{w \mid M_w \text{ hält für die leere Eingabe}\}$

Satz

Das Halteproblem auf leerem Band ist unentscheidbar.

Beweis: Wir reduzieren H auf H_0: Sei $f(w_M\#x) = w_{M_{0,x}}$, wobei TM $M_{0,x}$ erst x auf das Band schreibt, sich dann wie M verhält.

$$\begin{array}{ll} & w_M\#x \in H \\ \text{g.d.w.} & M_w \text{ hält für Eingabe } x \\ \text{g.d.w.} & M_{0,x} \text{ hält für die leere Eingabe} \\ \text{g.d.w.} & w_{M_{0,x}} \in H_0 \\ \text{g.d.w.} & f(w_M\#x) \in H_0 \end{array}$$

Funktion f kann durch eine TM berechnet werden. Daher gilt $H \leq H_0$. Da H unentscheidbar, ist H_0 unentscheidbar.

Quiz 7

Hochschule RheinMain

Sei $L \subseteq \Sigma^*$ und $L' \subseteq \Sigma'^*$. Wir schreiben $L' \leq L$, wenn

- Sprache L auf Sprache L' reduzierbar ist
- Sprache L' auf Sprache L reduzierbar ist
- Wenn es eine totale, berechenbare Funktion $f : \Sigma^* \to \Sigma'^*$ gibt mit $w \in L$ gdw. $f(w) \in L'$
- Wenn es eine totale, berechenbare Funktion $f : \Sigma'^* \to \Sigma^*$ gibt mit $w \in L'$ gdw. $f(w) \in L$

arsnova.hs-rm.de
6750 1376

Hochschule RheinMain

DAS POSTSCHE KORRESPONDENZPROBLEM

Das Postsche Korrespondenzproblem

Hochschule RheinMain

Motivation / Überblick

- Vorgeschlagen von Emil Post im Jahr 1946
- Es ist ein einfaches aber unentscheidbares Problem
- Es wird häufig verwendet, um es auf andere Probleme zu reduzieren und deren Unentscheidbarkeit zu zeigen
- Es hat nichts mit Turingmaschinen und deren Akzeptanzverhalten zu tun (im Gegensatz zu den verschiedenen Varianten vom Halteproblem)

Definition des Postschen Korrespondenzproblems

Hochschule RheinMain

Definition (Postsches Korrespondenzproblem)

Gegeben sei ein Alphabet Σ und eine

Folge von Wortpaaren $K = ((x_1, y_1), \ldots, (x_k, y_k))$ mit $x_i, y_i \in \Sigma^+$.

Das **Postsche Korrespondenzproblem (PCP)** ist die Frage, ob es für die gegebene Folge K eine Folge von Indizes $i_1, \ldots, i_m$ mit $i_j \in \{1, \ldots, k\}$ gibt, sodass

$$x_{i_1} \cdots x_{i_m} = y_{i_1} \cdots y_{i_m}$$

gilt.

PCP ist wie ein Domino-Spiel

Hochschule RheinMain

Spielstein**arten**: $(\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \ldots, \begin{bmatrix} x_k \\ y_k \end{bmatrix})$

Gesucht: Aneinandereihung der Spielsteine, sodass oben wie unten dasselbe Wort abgelesen werden kann. Dabei dürfen beliebig (aber endlich) viele Spielsteine verwendet werden.

Beispiel:

Sei $K = (\begin{bmatrix} a \\ aba \end{bmatrix}, \begin{bmatrix} baa \\ aa \end{bmatrix}, \begin{bmatrix} ab \\ bb \end{bmatrix})$

$I = (1, 2, 3, 2)$ ist eine Lösung, da

$$\begin{bmatrix} a \\ aba \end{bmatrix} \begin{bmatrix} baa \\ aa \end{bmatrix} \begin{bmatrix} ab \\ bb \end{bmatrix} \begin{bmatrix} baa \\ aa \end{bmatrix} = abaaabbaa$$
$$= abaaabbaa$$

PCP-Beispiel

Hochschule RheinMain

Instanz $K = (\begin{bmatrix} ab \\ bba \end{bmatrix}, \begin{bmatrix} ba \\ baa \end{bmatrix}, \begin{bmatrix} ba \\ aba \end{bmatrix}, \begin{bmatrix} bba \\ b \end{bmatrix})$

Die kürzeste Lösung benötigt 66 Paare:

(2, 1, 3, 1, 1, 2, 4, 2, 1, 3, 1, 3, 1, 1, 3, 1, 1, 2, 4, 1, 1, 2, 4, 3, 1, 4, 4, 3, 1, 1, 1, 2, 4, 2, 4, 4, 4, 3, 1, 3, 1, 4, 2, 4, 1, 1, 2, 4, 1, 4, 4, 3, 1, 4, 4, 3, 4, 4, 3, 4, 2, 4, 1, 4, 4, 3).

Unentscheidbarkeit PCP

Hochschule RheinMain

Satz

Das Postsche Korrespondenzproblem ist unentscheidbar.

Beweis: siehe Literatur

Anzahl k der Spielsteinarten beschränken

Hochschule RheinMain

PCP mit k-vielen verschiedenen Spielsteinarten:

- $k = 1$ oder $k = 2$: als entscheidbar gezeigt, im Jahr 1982
- $k \geq 5$: als unentscheidbar gezeigt im Jahr 2015
  (vorher war $k \geq 7$ bekannt (1996)
- $k = 3, 4$: unbekannt