Übung "Grundbegriffe der Informatik"

Karlsruher Institut für Technologie

Matthias Schulz, Gebäude 50.34, Raum 247

email: schulz@ira.uka.de

Matthias Janke, Gebäude 50.34, Raum 249

email: matthias.janke@kit.edu

Halteproblem - Variation

$prim(k)$ gibt wahr aus, falls k Primzahl ist, sonst falsch.

$n \leftarrow 4$;
$m \leftarrow 2$;
while $m < n$ **do**
  if $(prim(m) \wedge prim(n-m))$ **then**
    $n \leftarrow n + 2$
    $m \leftarrow 2$
  else
    $m \leftarrow m + 1$
  fi
od

.

Halteproblem - Variation

Programm hält, sobald es gerade Zahl ≥ 4 gefunden hat, die sich nicht als Summe zweier Primzahlen darstellen lässt.

.

Halteproblem - Variation

Programm hält, sobald es gerade Zahl ≥ 4 gefunden hat, die sich nicht als Summe zweier Primzahlen darstellen lässt.

Vermutung: Programm hält nie an!

.

Halteproblem - Variation

Programm hält, sobald es gerade Zahl ≥ 4 gefunden hat, die sich nicht als Summe zweier Primzahlen darstellen lässt.

Vermutung: Programm hält nie an!

Vermutung wurde 1742 geäußert

.

Halteproblem - Variation

Programm hält, sobald es gerade Zahl ≥ 4 gefunden hat, die sich nicht als Summe zweier Primzahlen darstellen lässt.

Vermutung: Programm hält nie an!

Vermutung wurde 1742 geäußert,
und bis heute nicht bestätigt/widerlegt!

(Siehe Goldbachsche Vermutung)

.

Unentscheidbarkeit - Beweise

Möglichkeit 1: Zeige: Falls Problem mit Turingmaschine entscheidbar ist, kann ich Turingmaschine bauen, die widersprüchlich ist.

.

Unentscheidbarkeit - Beweise

Möglichkeit 1: Zeige: Falls Problem mit Turingmaschine entscheidbar ist, kann ich Turingmaschine bauen, die widersprüchlich ist.

Beispiel: Falls es eine Turingmaschine gibt, die das Halteproblem entscheidet, gibt es auch eine Turingmaschine, die genau dann hält, wenn sie nicht hält.

.

Unentscheidbarkeit - Beweise

Möglichkeit 2 (schnell wachsende Funktionen): Zeige: Funktion wächst schneller als jede berechenbare Funktion, kann also nicht berechenbar sein

.

Unentscheidbarkeit - Beweise

Möglichkeit 2 (schnell wachsende Funktionen): Zeige: Funktion wächst schneller als jede berechenbare Funktion, kann also nicht berechenbar sein

Beispiel: Busy Beaver Funktion (mehr dazu später)

.

Unentscheidbarkeit - Beweise

Möglichkeit 3: Zeige: Wenn Problem mit Turingmaschine entscheidbar ist, dann kann ich algorithmisch ein bekanntes unentscheidbares Problem lösen.

.

Unentscheidbarkeit - Beweise

Halteproblem für leeres Band: Sprache aller Codierungen von Turingmaschinen, die für die Eingabe des leeren Wortes irgendwann anhalten.

.

Unentscheidbarkeit - Beweise

Busy Beaver nicht berechenbar $\Rightarrow$ Halteproblem für leeres Band unentscheidbar.

.

Unentscheidbarkeit - Beweise

Busy Beaver nicht berechenbar $\Rightarrow$ Halteproblem für leeres Band unentscheidbar.

Widerspruchsbeweis:

Angenommen, Halteproblem entscheidbar.

.

Unentscheidbarkeit - Beweise

Busy Beaver nicht berechenbar $\Rightarrow$ Halteproblem für leeres Band unentscheidbar.

Widerspruchsbeweis:

Angenommen, Halteproblem entscheidbar.

Dann kann ich für jedes $n \in \mathbb{N}_0$ $bb(n)$ berechnen!

.

Unentscheidbarkeit - Beweise

Vorgehen: Finde mit Hilfe des Algorithmus für Halteproblem haltende TM.

.

Unentscheidbarkeit - Beweise

Vorgehen: Finde mit Hilfe des Algorithmus für Halteproblem haltende TM.

Lass alle haltenden TM laufen.

.

Unentscheidbarkeit - Beweise

Vorgehen: Finde mit Hilfe des Algorithmus für Halteproblem haltende TM.

Lass alle haltenden TM laufen.

Höchste Anzahl von Einsen ist gesuchter Funktionswert.

.

Unentscheidbarkeit - Beweise

Da $bb(n)$ nicht berechenbar ist, ist Halteproblem für leeres Band unentscheidbar.

.

Unentscheidbarkeit - Beweise

Halteproblem für leeres Band unentscheidbar $\Rightarrow$ Äquivalenz von Turingmaschinen unentscheidbar

.

Unentscheidbarkeit - Beweise

Halteproblem für leeres Band unentscheidbar $\Rightarrow$ Äquivalenz von Turingmaschinen unentscheidbar

Widerspruchsbeweis:

Angenommen, Äquivalenz von Turingmaschinen ist entscheidbar.

.

Unentscheidbarkeit - Beweise

Halteproblem für leeres Band unentscheidbar $\Rightarrow$ Äquivalenz von Turingmaschinen unentscheidbar

Widerspruchsbeweis:

Angenommen, Äquivalenz von Turingmaschinen ist entscheidbar.

Dann kann ich für gegebene Turingmaschine T bestimmen, ob T bei Eingabe ϵ hält.

.

Unentscheidbarkeit - Beweise

Vorgehen: Konstruiere T' wie folgt:

- Falls erstes Zeichen $\square$, laufe unendlich nach rechts.

.

Unentscheidbarkeit - Beweise

Vorgehen: Konstruiere T' wie folgt:

- Falls erstes Zeichen $\square$, laufe unendlich nach rechts.
- Ansonsten mache das, was T macht.

.

Unentscheidbarkeit - Beweise

Vorgehen: Konstruiere T' wie folgt:

- Falls erstes Zeichen $\square$, laufe unendlich nach rechts.
- Ansonsten mache das, was T macht.

Für alle Wörter $\neq \epsilon$ machen T und T' das Gleiche.

.

Unentscheidbarkeit - Beweise

Vorgehen: Konstruiere T' wie folgt:

- Falls erstes Zeichen $\square$, laufe unendlich nach rechts.
- Ansonsten mache das, was T macht.

Für **alle** Wörter machen T und T' das Gleiche $\iff$ T und T' machen für ϵ das Gleiche ...

.

Unentscheidbarkeit - Beweise

Vorgehen: Konstruiere T' wie folgt:

- Falls erstes Zeichen $\square$, laufe unendlich nach rechts.
- Ansonsten mache das, was T macht.

Für **alle** Wörter machen T und T' das Gleiche $\iff$ T und T' machen für ϵ das Gleiche, d.h. T hält für ϵ nicht.

.

Unentscheidbarkeit - Beweise

Halteproblem für leeres Band unentscheidbar $\Rightarrow$ Äquivalenz von Turingmaschinen unentscheidbar

Widerspruchsbeweis:

Angenommen, Äquivalenz von Turingmaschinen ist entscheidbar.

Dann kann ich für gegebene Turingmaschine T bestimmen, ob T bei Eingabe ϵ hält:

Überprüfe mit Algorithmus, ob T und T' äquivalent sind; falls ja, hält T nicht.

.

Busy Beaver

Funktion f berechenbar:

Es gibt TM, die bei Eingabe $Repr_2(n)$ am Ende $Repr_2(f(n))$ auf Band schreibt.

.

Busy Beaver

Funktion f berechenbar:

Es gibt TM, die bei Eingabe $Repr_2(n)$ am Ende $Repr_2(f(n))$ auf Band schreibt.

Äquivalent:

Es gibt TM, die bei Eingabe $Repr_2(n)$ am Ende $1^{f(n)}$ auf Band schreibt.

.

Busy Beaver

Falls $bb(n)$ berechenbar, auch $bb(n) + 1$ berechenbar mit Turingmaschine T.

.

Busy Beaver

Falls $bb(n)$ berechenbar, auch $bb(n) + 1$ berechenbar mit Turingmaschine T.

Codiere Bandsymbole von T durch Blöcke aus $\square, 1$.

.

Busy Beaver

Falls $bb(n)$ berechenbar, auch $bb(n) + 1$ berechenbar mit Turingmaschine T.

Codiere Bandsymbole von T durch Blöcke aus $\square, 1$.

T' berechnet $bb(n) + 1$ mit Eingabe aus $\{\square, 1\}^*$

.

Busy Beaver

Turingmaschine T_n:

- Schreibe n in Binärdarstellung (codiert durch Blöcke) auf Band;

- simuliere T'.

.

Busy Beaver

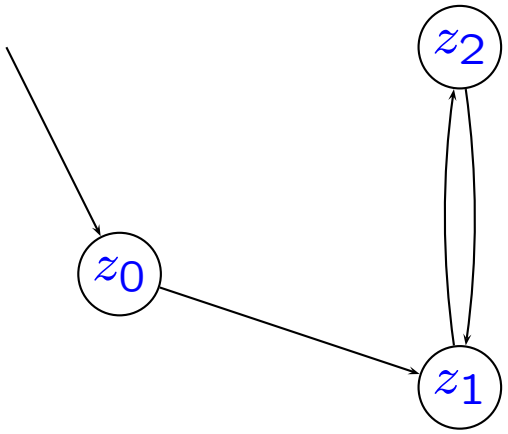
Turingmaschine T_n:

- Schreibe n in Binärdarstellung (codiert durch Blöcke) auf Band; $\Theta(\log n)$ *Zustände*

- simuliere T'. *konstant viele Zustände*

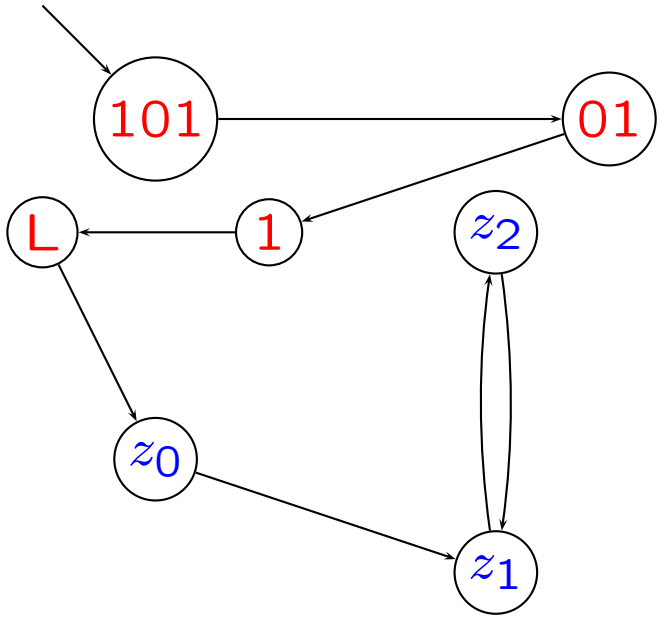
.

Busy Beaver



T'

Busy Beaver



Schreibe 101 auf Band, Mache, was T' macht

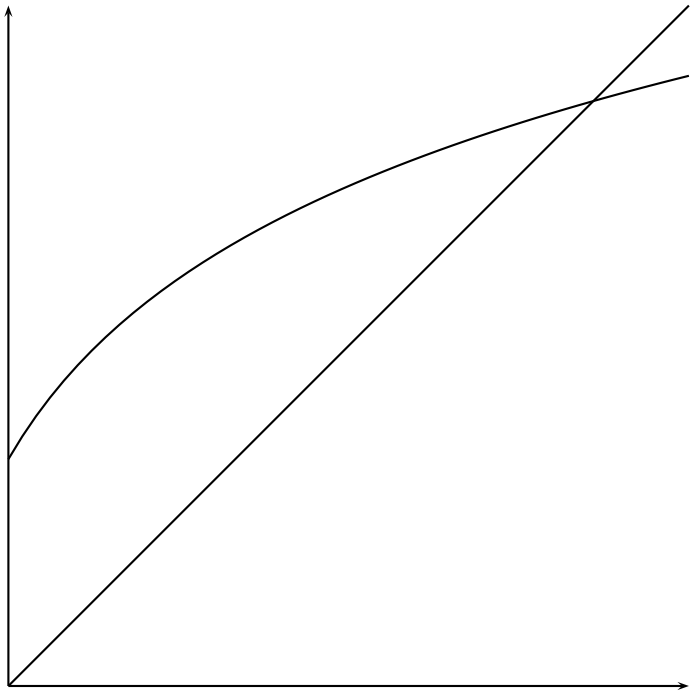
Busy Beaver

Falls $bb(n)$ berechenbar, auch $bb(n) + 1$ berechenbar mit Turingmaschine T'.

Anzahl Zustände von T_n: $k + c \log n$, k, c konstant.

.

Busy Beaver



Anzahl Zustände von T_n / n

Busy Beaver

Falls $bb(n)$ berechenbar, auch $bb(n) + 1$ berechenbar mit Turingmaschine T'.

Anzahl Zustände von T_n: $k + c \log n$, k, c konstant.

$\Rightarrow \exists n \in \mathbb{N}_0 : T_n$ hat höchstens n Zustände.

.

Busy Beaver

Falls $bb(n)$ berechenbar, auch $bb(n) + 1$ berechenbar mit Turingmaschine T'.

Anzahl Zustände von T_n: $k + c \log n$, k, c konstant.

$\Rightarrow \exists n \in \mathbb{N}_0 : T_n$ hat höchstens n Zustände.

$\Rightarrow$ Es gibt TM mit n Zuständen, die bei Eingabe von ϵ $bb(n) + 1$ Einsen auf Band schreibt.

.

Busy Beaver

Falls $bb(n)$ berechenbar, auch $bb(n) + 1$ berechenbar mit Turingmaschine T'.

Anzahl Zustände von T_n: $k + c \log n$, k, c konstant.

$\Rightarrow \exists n \in \mathbb{N}_0 : T_n$ hat höchstens n Zustände.

$\Rightarrow$ Es gibt TM mit n Zuständen, die bei Eingabe von ϵ $bb(n) + 1$ Einsen auf Band schreibt. **Widerspruch!**

.

Kontextfreie Grammatiken

Gegeben: KFG $G = (N, T, S, P)$.
Frage: Gilt $L(G) = T^*$?

.

Kontextfreie Grammatiken

Gegeben: KFG $G = (N, T, S, P)$.
Frage: Gilt $L(G) = T^*$?

Unentscheidbar!

.

Kontextfreie Grammatiken

Idee: Für Berechnung $c_0, c_1, \ldots c_k$, alle Konfigurationen gleiche Länge, schreibe Wort

$c_0 : c_1^R - c_2 : c_3^R \ldots (- \mid :)(c_k \mid c_k^R)$.

G erzeugt alle Wörter über Alphabet, die keine haltende Berechnung für Eingabe ϵ beschreiben.

.

Kontextfreie Grammatiken

Idee: Für Berechnung $c_0, c_1, \ldots c_k$, alle Konfigurationen gleiche Länge, schreibe Wort

$c_0 : c_1^R - c_2 : c_3^R \ldots (- \mid :)(c_k \mid c_k^R)$.

G erzeugt alle Wörter über Alphabet, die keine haltende Berechnung für Eingabe ϵ beschreiben.

Falls es keine haltende Berechnung gibt: Alle Wörter werden erzeugt!

.

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

-
-
-
-

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

- Zwischen : und − mehr oder weniger als ein Zustandssymbol,
-
-
-

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

- Zwischen : und − mehr oder weniger als ein Zustandssymbol,
- falsche Anfangs-/ Endkonfiguration,
-
-

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

- Zwischen : und − mehr oder weniger als ein Zustandssymbol,
- falsche Anfangs-/ Endkonfiguration,
- aufeinanderfolgende Konfigurationen entsprechen nicht Berechnung,
-

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

- Zwischen : und − mehr oder weniger als ein Zustandssymbol,
- falsche Anfangs-/ Endkonfiguration,
- aufeinanderfolgende Konfigurationen entsprechen nicht Berechnung,
- aufeinanderfolgende Konfigurationen nicht gleich lang.

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

Aufeinanderfolgende Konfigurationen entsprechen nicht Berechnung.

.

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

Aufeinanderfolgende Konfigurationen entsprechen nicht Berechnung.

$S \to XS_0X \mid XS_1X$
$X \to XX \mid \epsilon \mid \mathtt{a} \mid \mathtt{b} \mid \ldots$

.

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

Aufeinanderfolgende Konfigurationen entsprechen nicht Berechnung.

$S_0 \to \mathtt{a} S_0 \mathtt{a} \mid ...$

S_0 bedeutet: Fehler muss noch eingebaut werden, am Ende :!

.

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

Aufeinanderfolgende Konfigurationen entsprechen nicht Berechnung.

$S_1 \to \mathtt{a} S_1 \mathtt{a} \mid ...$

S_1 bedeutet: Fehler muss noch eingebaut werden, am Ende $-!$

.

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

Aufeinanderfolgende Konfigurationen entsprechen nicht Berechnung.

Fehler einbauen (Beispiel): Links steht $xsyz$, rechts steht Wort, das nicht zur Berechnung passt.

.

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

Aufeinanderfolgende Konfigurationen entsprechen nicht Berechnung.

Fehler einbauen (Beispiel): Links steht $xsyz$, rechts steht Wort, das nicht zur Berechnung passt.

Z.B. Mehr als ein Bandsymbol verändert, falsche Kopfbewegung, falscher Zustand etc.

.

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

Aufeinanderfolgende Konfigurationen entsprechen nicht Berechnung.

Nach eingebautem Fehler: Z_0 bzw Z_1 erzeugen links und rechts beliebige, gleich lange Wörter, ersetzen durch : bzw $-$.

.

Kontextfreie Grammatiken

Idee: Betrachte fehlerhafte Wörter:

Aufeinanderfolgende Konfigurationen entsprechen nicht Berechnung.

$Z_0 \to xZ_0y(x, y \in T)$ |:

.