Übung “Grundbegriffe der Informatik”

Karlsruher Institut für Technologie

Matthias Schulz, Gebäude 50.34, Raum 034

email: schulz@ira.uka.de

Halteproblem - Der Alternative Beweis

Gesucht: Turingmaschine T, die für Eingabe w in e_+ landet, falls

Turingmaschine T_w bei
Eingabe w

irgendwann anhält, sonst in e_-.

.

Halteproblem - Der Alternative Beweis

Gesucht: Turingmaschine T, die für Eingabe w in e_+ landet, falls

Turingmaschine T_w bei
Eingabe w

irgendwann anhält, sonst in e_-.

Angenommen, T existiert.

Erweitere Übergangsfunktion f, g, m :
$(e_+, x) \mapsto (e_+, x, 1)$,
um T' zu erhalten.

.

Halteproblem - Der Alternative Beweis

Also: Turingmaschine T' hält, falls sie in e_- landet, und geht in Endlosschleife, falls sie in e_+ landet.

.

Halteproblem - Der Alternative Beweis

Also: Turingmaschine T' hält, falls sie in e_- landet, und geht in Endlosschleife, falls sie in e_+ landet.

Also: T' hält bei Eingabe von w genau dann, wenn T_w bei Eingabe von w nicht anhält.

.

Halteproblem - Der Alternative Beweis

Also: Turingmaschine T' hält, falls sie in e_- landet, und geht in Endlosschleife, falls sie in e_+ landet.

Also: T' hält bei Eingabe von w genau dann, wenn T_w bei Eingabe von w **nicht** anhält.

.

Halteproblem - Der Alternative Beweis

Also: Turingmaschine T' hält, falls sie in e_- landet, und geht in Endlosschleife, falls sie in e_+ landet.

Also: T' hält bei Eingabe von w genau dann, wenn T_w bei Eingabe von w **nicht** anhält.

Preisfrage: Was macht T' bei Eingabe der eigenen Codierung?

.

Halteproblem - Der Alternative Beweis

Also: T' hält bei Eingabe von w genau dann, wenn T_w bei Eingabe von w **nicht** anhält.

Preisfrage: Was macht T' bei Eingabe der eigenen Codierung?

$T' = T_v$.

.

Halteproblem - Der Alternative Beweis

Also: T' hält bei Eingabe von w genau dann, wenn T_w bei Eingabe von w **nicht** anhält.

Preisfrage: Was macht T' bei Eingabe der eigenen Codierung?

$T' = T_v$.

Also: T_v hält bei Eingabe von v genau dann, wenn T_v bei Eingabe von v **nicht** anhält.

.

Halteproblem - Der Alternative Beweis

Damit muss die Annahme der Existenz von T falsch sein!

.

Äquivalenz zweier Turingmaschinen

Gegeben: Codierungen c_1, c_2 und (Codierung eines) Wort(es) w.

Gesucht: Turingmaschine, die überprüft, ob $T_{c_1}(w) = T_{c_2}(w)$.

.

Äquivalenz zweier Turingmaschinen

Gegeben: Codierungen c_1, c_2 und (Codierung eines) Wort(es) w.

Gesucht: Turingmaschine, die überprüft, ob $T_{c_1}(w) = T_{c_2}(w)$.

Annahme: T macht das!

.

Äquivalenz zweier Turingmaschinen

Gegeben: Codierungen c_1, c_2 und (Codierung eines) Wort(es) w.

Gesucht: Turingmaschine, die überprüft, ob $T_{c_1}(w) = T_{c_2}(w)$.

Annahme: T macht das!

Konstruiere Turingmaschine $I = (\{0\}, 0, X, f, g, m) = T_i$ mit

$\forall x \in X : (f, g, m)(0, x) = (0, x, 1)$

.

Äquivalenz zweier Turingmaschinen

Gegeben: Codierungen c_1, c_2 und (Codierung eines) Wort(es) w.

Gesucht: Turingmaschine, die überprüft, ob $T_{c_1}(w) = T_{c_2}(w)$.

Annahme: T macht das!

Konstruiere Turingmaschine $I = (\{0\}, 0, X, f, g, m) = T_i$ mit

$\forall x \in X : (f, g, m)(0, x) = (0, x, 1)$

Sei w beliebig.

Wende T auf Eingabe w, i, w an.

.

Äquivalenz zweier Turingmaschinen

Gegeben: Codierungen c_1, c_2 und (Codierung eines) Wort(es) w.

Gesucht: Turingmaschine, die überprüft, ob $T_{c_1}(w) = T_{c_2}(w)$.

Annahme: T macht das!

Sei w beliebig.

Wende T auf Eingabe w, i, w an.

T akzeptiert **genau dann**, wenn $T_w(w)$ **nicht** hält.

.

Äquivalenz zweier Turingmaschinen

$\bar{T}$: Vertausche akzeptierende/ablehnende Zustände:

$\bar{T}$ akzeptiert **genau dann**, wenn $T_w(w)$ hält.

.

Äquivalenz zweier Turingmaschinen

$\bar{T}$: Vertausche akzeptierende/ablehnende Zustände:

$\bar{T}$ akzeptiert **genau dann**, wenn $T_w(w)$ hält.

$\bar{T}$ entscheidet das Halteproblem!

.

Äquivalenz zweier Turingmaschinen

ACHTUNG!

.

Äquivalenz zweier Turingmaschinen

ACHTUNG!

Wir haben NICHT gezeigt, dass man das Halteproblem **doch** entscheiden kann!

.

Äquivalenz zweier Turingmaschinen

ACHTUNG!

Wir haben NICHT gezeigt, dass man das Halteproblem **doch** entscheiden kann!

Wir haben gezeigt, dass unsere Annahme (Turingmaschine, die überprüft, ob zwei Turingmaschinen bei fester Eingabe gleiches Ergebnis liefern) **falsch** war!

.

Äquivalenz zweier Turingmaschinen

Folgerung: Es gibt keine Turingmaschine, die für Codierungen w_1, w_2 entscheidet, ob zugehörige Turingmaschinen für **alle** Eingaben gleiche Ausgaben produzieren.

.

Äquivalenz zweier Turingmaschinen

Folgerung: Es gibt keine Turingmaschine, die für Codierungen w_1, w_2 entscheidet, ob zugehörige Turingmaschinen für **alle** Eingaben gleiche Ausgaben produzieren.

Konstruktion: Bastle Turingmaschinen $T^{w_1,w}, T^{w_2,w}$, die

.

Äquivalenz zweier Turingmaschinen

Folgerung: Es gibt keine Turingmaschine, die für Codierungen w_1, w_2 entscheidet, ob zugehörige Turingmaschinen für **alle** Eingaben gleiche Ausgaben produzieren.

Konstruktion: Bastle Turingmaschinen $T^{w_1,w}, T^{w_2,w}$, die

- zuerst überprüfen, ob die Eingabe w ist,

.

Äquivalenz zweier Turingmaschinen

Folgerung: Es gibt keine Turingmaschine, die für Codierungen w_1, w_2 entscheidet, ob zugehörige Turingmaschinen für **alle** Eingaben gleiche Ausgaben produzieren.

Konstruktion: Bastle Turingmaschinen $T^{w_1,w}, T^{w_2,w}$, die

- zuerst überprüfen, ob die Eingabe w ist,
- falls nicht, in eine Endlosschleife übergehen,

.

Äquivalenz zweier Turingmaschinen

Folgerung: Es gibt keine Turingmaschine, die für Codierungen w_1, w_2 entscheidet, ob zugehörige Turingmaschinen für **alle** Eingaben gleiche Ausgaben produzieren.

Konstruktion: Bastle Turingmaschinen $T^{w_1,w}, T^{w_2,w}$, die

- zuerst überprüfen, ob die Eingabe w ist,
- falls nicht, in eine Endlosschleife übergehen,
- falls doch, T_{w_1} bzw. T_{w_2} simulieren.

.

Äquivalenz zweier Turingmaschinen

Folgerung: Es gibt keine Turingmaschine, die für Codierungen w_1, w_2 entscheidet, ob zugehörige Turingmaschinen für **alle** Eingaben gleiche Ausgaben produzieren.

$T^{w_1,w}$ und $T^{w_2,w}$ verhalten sich für alle Eingaben außer w gleich.

.

Äquivalenz zweier Turingmaschinen

Folgerung: Es gibt keine Turingmaschine, die für Codierungen w_1, w_2 entscheidet, ob zugehörige Turingmaschinen für **alle** Eingaben gleiche Ausgaben produzieren.

$T^{w_1,w}$ und $T^{w_2,w}$ verhalten sich für alle Eingaben außer w gleich.

$T^{w_1,w}$ und $T^{w_2,w}$ verhalten sich für alle Eingaben genau dann gleich, falls sie sich für w gleich verhalten.

.

Äquivalenz zweier Turingmaschinen

Folgerung: Es gibt keine Turingmaschine, die für Codierungen w_1, w_2 entscheidet, ob zugehörige Turingmaschinen für **alle** Eingaben gleiche Ausgaben produzieren.

$T^{w_1,w}$ und $T^{w_2,w}$ verhalten sich für alle Eingaben außer w gleich.

$T^{w_1,w}$ und $T^{w_2,w}$ verhalten sich für alle Eingaben genau dann gleich, falls sich T_{w_1} und T_{w_2}für w gleich verhalten.

.

Äquivalenz zweier Turingmaschinen

Folgerung: Es gibt keine Turingmaschine, die für Codierungen w_1, w_2 entscheidet, ob zugehörige Turingmaschinen für **alle** Eingaben gleiche Ausgaben produzieren.

$T^{w_1,w}$ und $T^{w_2,w}$ verhalten sich für alle Eingaben außer w gleich.

$T^{w_1,w}$ und $T^{w_2,w}$ verhalten sich für alle Eingaben genau dann gleich, falls sich T_{w_1} und T_{w_2}für w gleich verhalten.

Da das unentscheidbar ist, ist auch das “größere” Problem unentscheidbar.

.

Busy-Beaver DELUXE

Für $n, k \in \mathbb{N}_0, w \in \{0, 1, x\}^*$ sei $bbd(n, k, w)$ definiert als die größte Zahl von Einsen, die eine **anhaltende** Turingmaschine mit

- $n + 1$ Zuständen,
- $k + 3$ Symbolen (unter denen sich $0, 1, x$ befinden)
- bei Eingabe von w

am Ende auf das Band geschrieben haben kann.

.

Busy-Beaver DELUXE

Behauptung: Für jede berechenbare Funktion $F(n, k, w)$ gibt es Tripel (n, k, w) mit $bbd(n, k, w) > F(n, k, w)$.

.

Busy-Beaver DELUXE

Idee: Wenn $F(n, k, w)$ berechenbar, dann auch $G(n, k, w) = 2^{F(n,k,w)+1} - 1$.

.

Busy-Beaver DELUXE

Idee: Wenn $F(n,k,w)$ berechenbar, dann auch $G(n,k,w) = 2^{F(n,k,w)+1} - 1$.

Turingmaschine T berechnet $G(n,k,w)$ bei Eingabe des Wortes

$Repr_2(n)xRepr_2(k)xw$,

das heißt, am Ende steht $Repr_2(G(n,k,w))$ auf dem Band.

.

Busy-Beaver DELUXE

Idee: Wenn $F(n,k,w)$ berechenbar, dann auch $G(n,k,w) = 2^{F(n,k,w)+1} - 1$.

Turingmaschine T berechnet $G(n,k,w)$ bei Eingabe des Wortes

$Repr_2(n)xRepr_2(k)xw$.

Turingmaschine T' macht aus Eingabe $Repr_2(n)xRepr_2(k)xw$ zuerst $Repr_2(n)xRepr_2(k)xRepr_2(n)xRepr_2(k)xw$ und simuliert dann T.

.

Busy-Beaver DELUXE

Idee: Wenn $F(n,k,w)$ berechenbar, dann auch $G(n,k,w) = 2^{F(n,k,w)+1} - 1$.

Turingmaschine T berechnet $G(n,k,w)$ bei Eingabe des Wortes

$Repr_2(n)xRepr_2(k)xw$.

Turingmaschine T' macht aus Eingabe $Repr_2(n)xRepr_2(k)xw$ zuerst $Repr_2(n)xRepr_2(k)xRepr_2(n)xRepr_2(k)xw$ und simuliert dann T.

T' habe $n'+1$ Zustände und $k'+3$ Symbole.

.

Busy-Beaver DELUXE

Idee: Wenn $F(n, k, w)$ berechenbar, dann auch $G(n, k, w) = 2^{F(n,k,w)+1} - 1$.

Turingmaschine T berechnet $G(n, k, w)$ bei Eingabe des Wortes

$Repr_2(n)xRepr_2(k)xw$.

T' habe $n' + 1$ Zustände und $k' + 3$ Symbole.

Betrachte T' bei Eingabe von $Repr_2(n')xRepr_2(k')x$.

.

Busy-Beaver DELUXE

Betrachte T' bei Eingabe von $w = Repr_2(n')xRepr_2(k')x$.

T' ändert Eingabe zu $Repr_2(n')xRepr_2(k')xw$ und simuliert dann T.

.

Busy-Beaver DELUXE

Betrachte T' bei Eingabe von $w = Repr_2(n')xRepr_2(k')x$.

T' ändert Eingabe zu $Repr_2(n')xRepr_2(k')xw$ und simuliert dann T.

T gibt $Repr_2(G(n', k', w))$ aus.

.

Busy-Beaver DELUXE

Betrachte T' bei Eingabe von $w = Repr_2(n')xRepr_2(k')x$.

T' ändert Eingabe zu $Repr_2(n')xRepr_2(k')xw$ und simuliert dann T.

T gibt $Repr_2(G(n', k', w))$ aus.

Also: T' schreibt das Wort $Repr_2(2^{F(n',k',w)+1} - 1)$ auf das Band.

.

Busy-Beaver DELUXE

Betrachte T' bei Eingabe von $w = Repr_2(n')xRepr_2(k')x$.

T' ändert Eingabe zu $Repr_2(n')xRepr_2(k')xw$ und simuliert dann T.

T gibt $Repr_2(G(n',k',w))$ aus.

Also: T' schreibt das Wort $1^{F(n',k',w)+1}$ auf das Band.

.

Busy-Beaver DELUXE

Betrachte T' bei Eingabe von $w = Repr_2(n')xRepr_2(k')x$.

T' ändert Eingabe zu $Repr_2(n')xRepr_2(k')xw$ und simuliert dann T.

T gibt $Repr_2(G(n', k', w))$ aus.

Also: T' schreibt das Wort $1^{F(n',k',w)+1}$ auf das Band.

Da T' $n' + 1$ Zustände und $k' + 3$ Symbole hat, gilt

$bbd(n', k', w) \geq$ Anzahl der Einsen, die T' bei Eingabe von w auf Band schreibt.

.

Busy-Beaver DELUXE

Betrachte T' bei Eingabe von $w = Repr_2(n')xRepr_2(k')x$.

T' schreibt das Wort $1^{F(n',k',w)+1}$ auf das Band.

Da T' $n'+1$ Zustände und $k'+3$ Symbole hat, gilt

$bbd(n', k', w) \geq$ Anzahl der Einsen, die T' bei Eingabe von w auf Band schreibt.

Also gilt $bbd(n', k', w) \geq F(n', k', w) + 1$.

.

Busy-Beaver DELUXE

Da sich $bbd(n, k, w)$ von jeder berechenbaren Funktion für mindestens ein Tripel (n, k, w) unterscheidet, kann bbd nicht berechenbar sein.

.