Vorlesung

Grundlagen der Theoretischen Informatik / Einführung in die Theoretische Informatik I

Bernhard Beckert

Institut für Informatik

UNIVERSITÄT KOBLENZ · LANDAU

Sommersemester 2007

Dank

Diese Vorlesungsmaterialien basieren ganz wesentlich auf den Folien zu den Vorlesungen von

Katrin Erk (gehalten an der Universität Koblenz-Landau)

Jürgen Dix (gehalten an der TU Clausthal)

Ihnen beiden gilt mein herzlicher Dank.

– Bernhard Beckert, April 2007

Indeterminierte Turing-Maschine

Beispiel 11.3

Eine indeterminierte Turing-Maschine, die

$$L = \{w \in \{a,b\}^* \mid w \text{ besitzt } aba \text{ als Teilwort}\}$$

azkeptiert.

Siehe Tafel.

Indeterminierte Turing-Maschine

Beispiel 11.4

Sei

$$L = \{|^n \mid n \text{ ist nicht prim und } n \geq 2\}$$

Eine NTM kann diese Sprache wie folgt azeptieren:

1. Eine Zahl „raten“ und (nach rechts) aufs Band schreiben.
2. Noch eine Zahl „raten“ und daneben schreiben.
3. Die beiden Zahlen miteinander multiplizieren.
4. Das Ergebnis mit der Eingabe vergleichen.
5. Genau dann, wenn beide gleich sind, anhalten

Indeterminierte Turing-Maschine

Beispiel 11.4

Sei

$$L = \{|^n \mid n \text{ ist nicht prim und } n \geq 2\}$$

Eine NTM kann diese Sprache wie folgt azeptieren:

1. Eine Zahl „raten“ und (nach rechts) aufs Band schreiben.
2. Noch eine Zahl „raten“ und daneben schreiben.
3. Die beiden Zahlen miteinander multiplizieren.
4. Das Ergebnis mit der Eingabe vergleichen.
5. Genau dann, wenn beide gleich sind, anhalten

Indeterminierte Turing-Maschine

Beispiel 11.4

Sei

$$L = \{|^n \mid n \text{ ist nicht prim und } n \geq 2\}$$

Eine NTM kann diese Sprache wie folgt azeptieren:

1. Eine Zahl „raten“ und (nach rechts) aufs Band schreiben.
2. Noch eine Zahl „raten“ und daneben schreiben.
3. Die beiden Zahlen miteinander multiplizieren.
4. Das Ergebnis mit der Eingabe vergleichen.
5. Genau dann, wenn beide gleich sind, anhalten

Indeterminierte Turing-Maschine

Beispiel 11.4

Sei

$$L = \{|^n \mid n \text{ ist nicht prim und } n \geq 2\}$$

Eine NTM kann diese Sprache wie folgt azeptieren:

1. Eine Zahl „raten“ und (nach rechts) aufs Band schreiben.
2. Noch eine Zahl „raten“ und daneben schreiben.
3. Die beiden Zahlen miteinander multiplizieren.
4. Das Ergebnis mit der Eingabe vergleichen.
5. Genau dann, wenn beide gleich sind, anhalten

Indeterminierte Turing-Maschine

Beispiel 11.4

Sei

$$L = \{|^n \mid n \text{ ist nicht prim und } n \geq 2\}$$

Eine NTM kann diese Sprache wie folgt azeptieren:

1. Eine Zahl „raten“ und (nach rechts) aufs Band schreiben.
2. Noch eine Zahl „raten“ und daneben schreiben.
3. Die beiden Zahlen miteinander multiplizieren.
4. Das Ergebnis mit der Eingabe vergleichen.
5. Genau dann, wenn beide gleich sind, anhalten

Indeterminierte Turing-Maschine

Beispiel 11.4

Sei

$$L = \{|^n \mid n \text{ ist nicht prim und } n \geq 2\}$$

Eine NTM kann diese Sprache wie folgt azeptieren:

1. Eine Zahl „raten“ und (nach rechts) aufs Band schreiben.
2. Noch eine Zahl „raten“ und daneben schreiben.
3. Die beiden Zahlen miteinander multiplizieren.
4. Das Ergebnis mit der Eingabe vergleichen.
5. Genau dann, wenn beide gleich sind, anhalten

Indeterminierte Turing-Maschine

Theorem 11.5 (Simulation von NTM durch DTM)

Jede Sprache, die von einer indeterminierten Turing-Maschine akzeptiert wird, wird auch von einer Standard-DTM akzeptiert.

Beweis (Anfang)

Sei

- L eine Sprache über Σ_0^* mit $\# \notin \Sigma_0$;
- $\mathcal{M} = (K, \Sigma, \Delta, s)$ eine indeterminierte Turing-Maschine, die L akzeptiert.

Indeterminierte Turing-Maschine

Theorem 11.5 (Simulation von NTM durch DTM)

Jede Sprache, die von einer indeterminierten Turing-Maschine akzeptiert wird, wird auch von einer Standard-DTM akzeptiert.

Beweis (Anfang)

Sei

- L eine Sprache über Σ_0^* mit $\# \notin \Sigma_0$;
- $\mathcal{M} = (K, \Sigma, \Delta, s)$ eine indeterminierte Turing-Maschine, die L akzeptiert.

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Wir konstruieren zu $\mathcal{M}$ eine Standard-DTM $\mathcal{M}'$, die so rechnet:

- $\mathcal{M}'$ durchläuft systematisch **alle** Rechnungen von $\mathcal{M}$, und sucht dabei nach einer Haltekonfiguration.
- $\mathcal{M}'$ genau dann, wenn sie eine Haltekonfiguration von $\mathcal{M}$ findet.

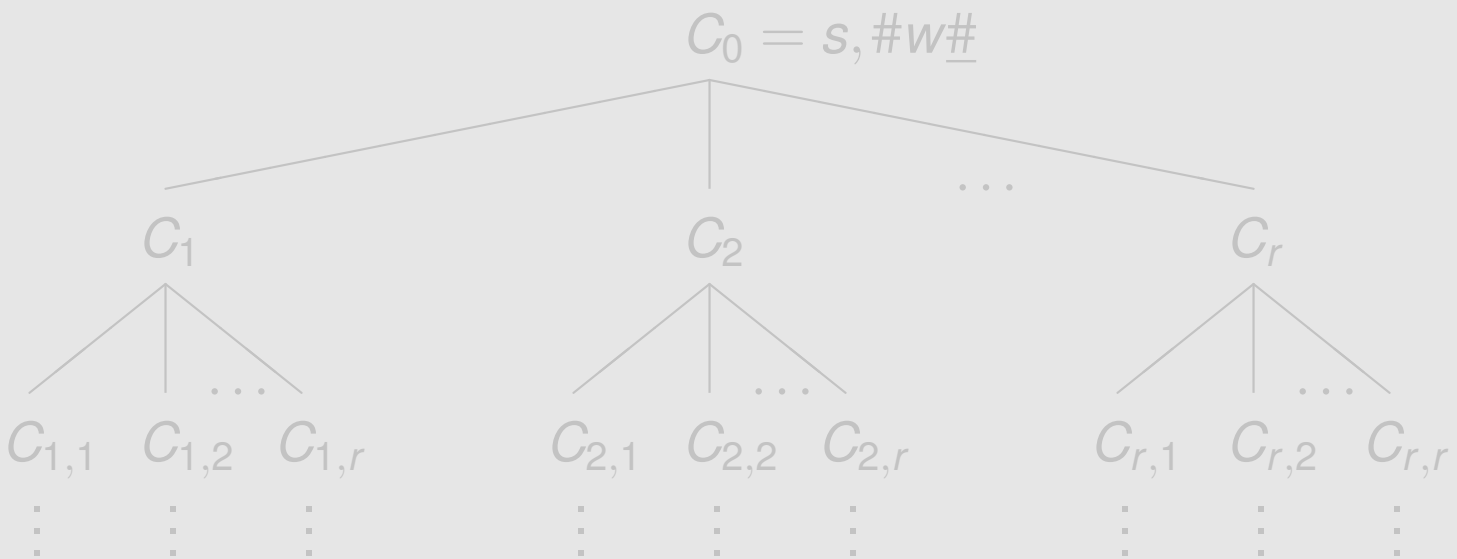
Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Suchbaum, Rechnungsbaum:

Stelle alle Rechnungen von $\mathcal{M}$ von einer Startkonfiguration C_0 dar als einen Baum mit Wurzel C_0.

Ein Ast ist eine mögliche Rechnung von $\mathcal{M}$.

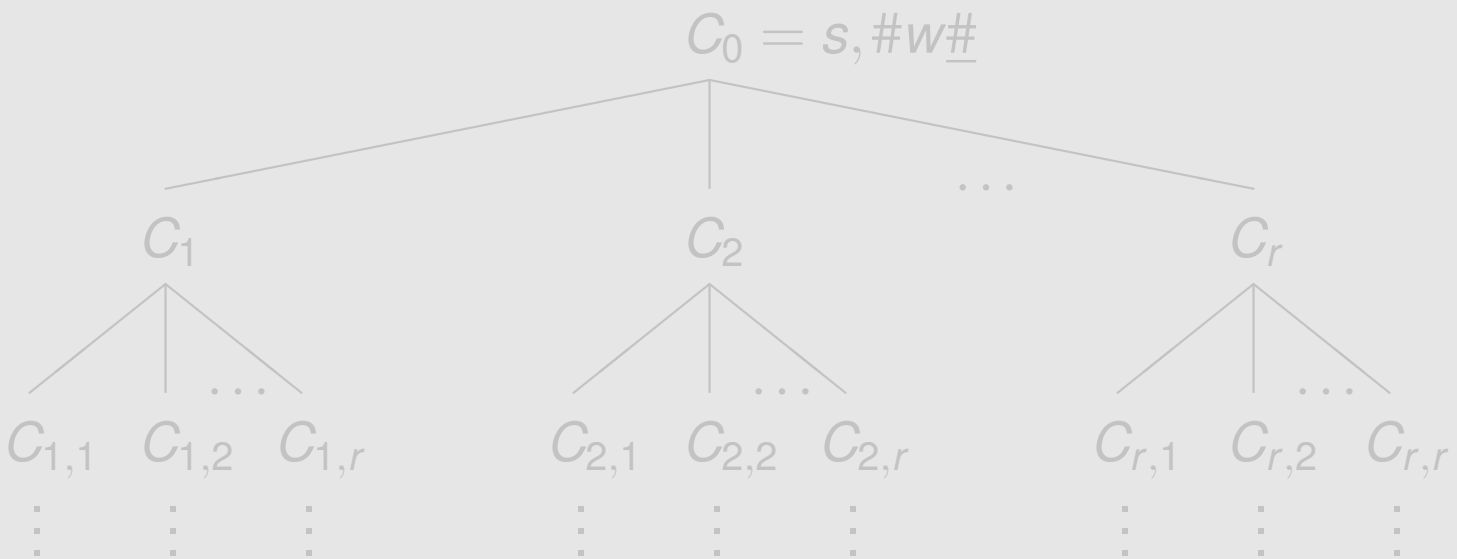


Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Suchbaum, Rechnungsbaum:
Stelle alle Rechnungen von $\mathcal{M}$ von einer Startkonfiguration C_0 dar als einen Baum mit Wurzel C_0.

Ein Ast ist eine mögliche Rechnung von $\mathcal{M}$.



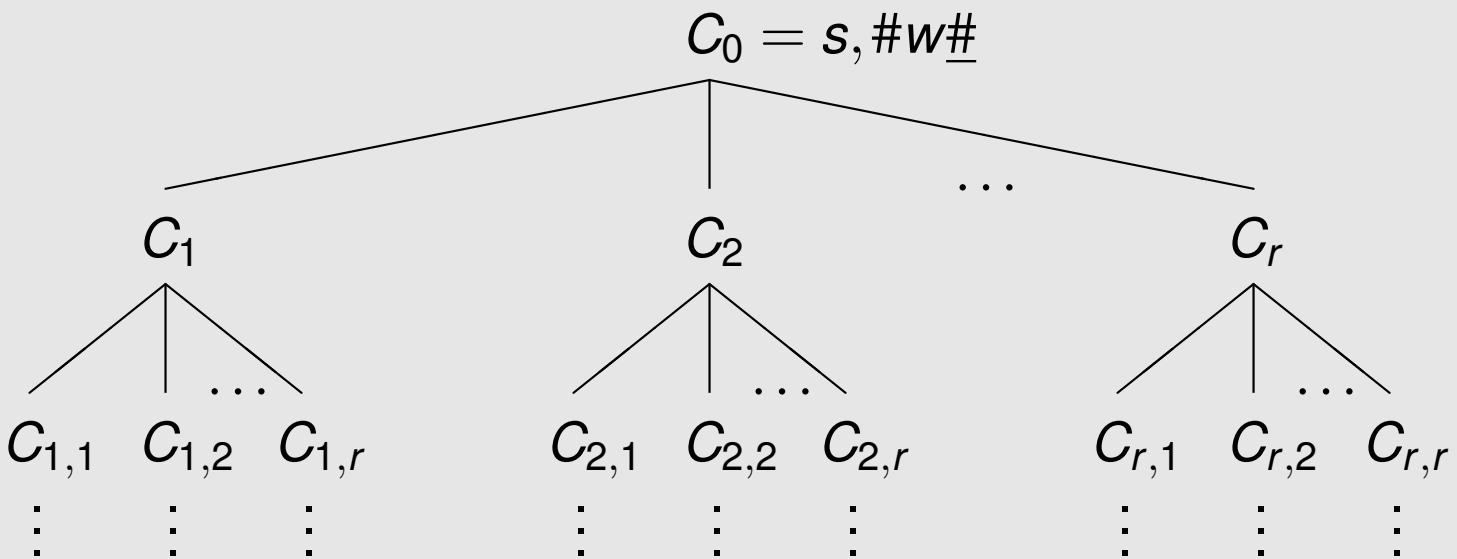
Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Suchbaum, Rechnungsbaum:

Stelle alle Rechnungen von $\mathcal{M}$ von einer Startkonfiguration C_0 dar als einen Baum mit Wurzel C_0.

Ein Ast ist eine mögliche Rechnung von $\mathcal{M}$.



Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Problem:

Es kann zur Startkonfiguration

$$C_0 = s, \#w\#\underline{\#}$$

unendlich viele Rechnungen von $\mathcal{M}$ geben,
und jede einzelne von ihnen kann unendlich lang sein.

Wir können also nicht erst einen Ast ganz durchlaufen und dann den nächsten Ast durchsuchen.

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Problem:

Es kann zur Startkonfiguration

$$C_0 = s, \#w\underline{\#}$$

unendlich viele Rechnungen von $\mathcal{M}$ geben,
und jede einzelne von ihnen kann unendlich lang sein.

Wir können also nicht erst einen Ast ganz durchlaufen und dann den nächsten Ast durchsuchen.

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Die Lösung:

Breitensuche

Drchlaufe den Rechnungsbaum nicht depth-first, sondern per **iterative deepening**.

- Untersuchen alle möglichen Rechnungen bis zum ersten Schritt.
- Untersuchen alle möglichen Rechnungen bis zum zweiten Schritt.
- Untersuchen alle möglichen Rechnungen bis zum dritten Schritt.
- usw.

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Die Lösung:

Breitensuche

Drchlaufe den Rechnungsbaum nicht depth-first, sondern per **iterative deepening**.

- Untersuchen alle möglichen Rechnungen bis zum ersten Schritt.
- Untersuchen alle möglichen Rechnungen bis zum zweiten Schritt.
- Untersuchen alle möglichen Rechnungen bis zum dritten Schritt.
- usw.

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Die Lösung:
Breitensuche

Drchlaufe den Rechnungsbaum nicht depth-first, sondern per **iterative deepening**.

- Untersuchen alle möglichen Rechnungen bis zum ersten Schritt.
- Untersuchen alle möglichen Rechnungen bis zum zweiten Schritt.
- Untersuchen alle möglichen Rechnungen bis zum dritten Schritt.
- usw.

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Können wir damit denn in endlicher Zeit eine Haltekonfiguration finden, falls es eine gibt?

Problem:
Kann der Rechnungsbaum nicht nur **unendlich tief**,
sondern auch **unendlich breit** werden?

Nein, denn:
Maximale Anzahl von Nachfolgekonfigurationen

$$r = max\{|\Delta(q,a)| \quad | \quad q \in K, a \in \Sigma\}$$

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Können wir damit denn in endlicher Zeit eine Haltekonfiguration finden, falls es eine gibt?

Problem:

Kann der Rechnungsbaum nicht nur **unendlich tief**,
sondern auch **unendlich breit** werden?

Nein, denn:

Maximale Anzahl von Nachfolgekonfigurationen

$$r = max\{|\Delta(q,a)| \quad | \quad q \in K, a \in \Sigma\}$$

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Können wir damit denn in endlicher Zeit eine Haltekonfiguration finden, falls es eine gibt?

Problem:
Kann der Rechnungsbaum nicht nur **unendlich tief**,
sondern auch **unendlich breit** werden?

Nein, denn:
Maximale Anzahl von Nachfolgekonfigurationen

$$r = max\{|\Delta(q,a)| \quad | \quad q \in K, a \in \Sigma\}$$

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

$\mathcal{M}'$ kann (z.B.) als eine 3-DTM gewählt werden:

- **Auf dem ersten Band steht immer das Eingabewort w.**
  Da die Rechnung immer wieder neu mit $s, \#w\underline{\#}$ von $\mathcal{M}$ beginnt, wird das Eingabewort immer wieder gebraucht.
- **Auf dem zweiten Band steht, welcher Weg durch den Rechnungsbaum gerade verfolgt wird.**
  Der Einfachheit halber: Wenn eine Konfiguration weniger als r Nachfolgekonfigurationen hat, soll der zugehörige Knoten trotzdem r Söhne haben, und die überzähligen Konfigurationen sind leer.

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

$\mathcal{M}'$ kann (z.B.) als eine 3-DTM gewählt werden:

- **Auf dem ersten Band steht immer das Eingabewort w.**
  Da die Rechnung immer wieder neu mit $s, \#w\underline{\#}$ von $\mathcal{M}$ beginnt, wird das Eingabewort immer wieder gebraucht.
- **Auf dem zweiten Band steht, welcher Weg durch den Rechnungsbaum gerade verfolgt wird.**
  Der Einfachheit halber: Wenn eine Konfiguration weniger als r Nachfolgekonfigurationen hat, soll der zugehörige Knoten trotzdem r Söhne haben, und die überzähligen Konfigurationen sind leer.

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Darstellung des aktuellen Pfades im Rechnungsbaum als Zahl im r-adischen System.

Eine Zahl $d_1 \dots d_n$ bedeutet:

- Von der Startkonfiguration C_0 aus ist die d_1-te der r möglichen Nachfolgekonfigurationen gewählt worden, C_{d_1}.
- Von C_{d_1}, einem Knoten der Tiefe 1, aus wurde die d_2-te mögliche Nachfolgekonfiguration gewählt,
- usw.

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Darstellung des aktuellen Pfades im Rechnungsbaum als Zahl im r-adischen System.

Eine Zahl $d_1 \ldots d_n$ bedeutet:

- Von der Startkonfiguration C_0 aus ist die d_1-te der r möglichen Nachfolgekonfigurationen gewählt worden, C_{d_1}.
- Von C_{d_1}, einem Knoten der Tiefe 1, aus wurde die d_2-te mögliche Nachfolgekonfiguration gewählt,
- usw.

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Ausführung des Iterative Deepening:

- Beginne mit 0 auf zweitem Band.
- Jeweils nächste zu betrachtende Rechnung erhöhen der Zahl auf Band 2 um 1
- Auf Band 3 wird eine Rechnung von $\mathcal{M}$ **determiniert** simuliert. Und zwar entsprechend der Zahl $d_1 \dots d_n$ auf Band 2. Die Endkonfiguration $C_{d_1 \dots d_n}$ dieser Rechnung steht im Rechnungsbaum an dem Knoten, der das Ende des Pfades $d_1 \dots d_n$ bildet.
- Ist die Konfiguration $C_{d_1 \dots d_n}$ eine Haltekonfiguration, so hält $\mathcal{M}'$.
- Sonst Zahl auf Band 2 erhöhen und die nächste Rechnungssimulation beginnen

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Ausführung des Iterative Deepening:

- Beginne mit 0 auf zweitem Band.
- Jeweils nächste zu betrachtende Rechnung erhöhen der Zahl auf Band 2 um 1
- Auf Band 3 wird eine Rechnung von $\mathcal{M}$ **determiniert** simuliert. Und zwar entsprechend der Zahl $d_1 \dots d_n$ auf Band 2. Die Endkonfiguration $C_{d_1 \dots d_n}$ dieser Rechnung steht im Rechnungsbaum an dem Knoten, der das Ende des Pfades $d_1 \dots d_n$ bildet.
- Ist die Konfiguration $C_{d_1 \dots d_n}$ eine Haltekonfiguration, so hält $\mathcal{M}'$.
- Sonst Zahl auf Band 2 erhöhen und die nächste Rechnungssimulation beginnen

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Ausführung des Iterative Deepening:

- Beginne mit 0 auf zweitem Band.
- Jeweils nächste zu betrachtende Rechnung erhöhen der Zahl auf Band 2 um 1
- Auf Band 3 wird eine Rechnung von $\mathcal{M}$ **determiniert** simuliert. Und zwar entsprechend der Zahl $d_1 \dots d_n$ auf Band 2. Die Endkonfiguration $C_{d_1 \dots d_n}$ dieser Rechnung steht im Rechnungsbaum an dem Knoten, der das Ende des Pfades $d_1 \dots d_n$ bildet.
- Ist die Konfiguration $C_{d_1 \dots d_n}$ eine Haltekonfiguration, so hält $\mathcal{M}'$.
- Sonst Zahl auf Band 2 erhöhen und die nächste Rechnungssimulation beginnen

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Ausführung des Iterative Deepening:

- Beginne mit 0 auf zweitem Band.
- Jeweils nächste zu betrachtende Rechnung erhöhen der Zahl auf Band 2 um 1
- Auf Band 3 wird eine Rechnung von $\mathcal{M}$ **determiniert** simuliert. Und zwar entsprechend der Zahl $d_1 \ldots d_n$ auf Band 2. Die Endkonfiguration $C_{d_1 \ldots d_n}$ dieser Rechnung steht im Rechnungsbaum an dem Knoten, der das Ende des Pfades $d_1 \ldots d_n$ bildet.
- Ist die Konfiguration $C_{d_1 \ldots d_n}$ eine Haltekonfiguration, so hält $\mathcal{M}'$.
- Sonst Zahl auf Band 2 erhöhen und die nächste Rechnungssimulation beginnen

Indeterminierte Turing-Maschine

Beweis (Fortsetzung)

Ausführung des Iterative Deepening:

- Beginne mit 0 auf zweitem Band.
- Jeweils nächste zu betrachtende Rechnung erhöhen der Zahl auf Band 2 um 1
- Auf Band 3 wird eine Rechnung von $\mathcal{M}$ **determiniert** simuliert. Und zwar entsprechend der Zahl $d_1 \dots d_n$ auf Band 2. Die Endkonfiguration $C_{d_1 \dots d_n}$ dieser Rechnung steht im Rechnungsbaum an dem Knoten, der das Ende des Pfades $d_1 \dots d_n$ bildet.
- Ist die Konfiguration $C_{d_1 \dots d_n}$ eine Haltekonfiguration, so hält $\mathcal{M}'$.
- Sonst Zahl auf Band 2 erhöhen und die nächste Rechnungssimulation beginnen

Indeterminierte Turing-Maschine

Beweis (Ende)

Damit gilt:

$\mathcal{M}'$ **hält bei Input** w

gdw

es gibt in R_{C_0} **eine Haltekonfiguration.**

Das ist genau dann der Fall, wenn

- $\mathcal{M}$ bei Input w hält,
- w in L liegt.

□

Indeterminierte Turing-Maschine

Beweis (Ende)

Damit gilt:

$\mathcal{M}'$ **hält bei Input** w

gdw

es gibt in R_{C_0} **eine Haltekonfiguration.**

Das ist genau dann der Fall, wenn

- $\mathcal{M}$ bei Input w hält,
- w in L liegt.

□

Teil V

Teil V

Universelle Turing-Maschine

Vergleich Turing-Maschine / „normaler“ Computer

Turing-Maschinen sind sehr mächtig.

Wie mächtig sind sie wirklich?

- Eine Turing-Maschine hat eine vorgegebenes „Programm“ (Regelmenge)
- „Normale“ Computer können beliebige Programme ausführen.

Tatsächlich geht das mit Turing-Maschinen auch!

Universelle Turing-Maschine

Vergleich Turing-Maschine / „normaler“ Computer

Turing-Maschinen sind sehr mächtig.

Wie mächtig sind sie wirklich?

- Eine Turing-Maschine hat eine vorgegebenes „Programm“ (Regelmenge)
- „Normale“ Computer können beliebige Programme ausführen.

Tatsächlich geht das mit Turing-Maschinen auch!

Universelle Turing-Maschine

Vergleich Turing-Maschine / „normaler“ Computer

Turing-Maschinen sind sehr mächtig.

Wie mächtig sind sie wirklich?

- Eine Turing-Maschine hat eine vorgegebenes „Programm“ (Regelmenge)
- „Normale“ Computer können beliebige Programme ausführen.

Tatsächlich geht das mit Turing-Maschinen auch!

Universelle Turing-Maschine

Turing-Maschine, die andere TMen simuliert

- Universelle TM $\mathcal{U}$ bekommt als Eingabe:
  - die Regelmenge einer beliebigen Turing-Maschine $\mathcal{M}$ und
  - ein Wort w, auf dem $\mathcal{M}$ rechnen soll.
- $\mathcal{U}$ simuliert $\mathcal{M}$, indem sie jeweils nachschlägt, welchen δ-Übergang $\mathcal{M}$ machen würde.

Universelle Turing-Maschine

Turing-Maschine, die andere TMen simuliert

- Universelle TM $\mathcal{U}$ bekommt als Eingabe:
  - die Regelmenge einer beliebigen Turing-Maschine $\mathcal{M}$ und
  - ein Wort w, auf dem $\mathcal{M}$ rechnen soll.
- $\mathcal{U}$ simuliert $\mathcal{M}$, indem sie jeweils nachschlägt, welchen δ-Übergang $\mathcal{M}$ machen würde.

Universelle Turing-Maschine

Turing-Maschine, die andere TMen simuliert

- Universelle TM $\mathcal{U}$ bekommt als Eingabe:
  - die Regelmenge einer beliebigen Turing-Maschine $\mathcal{M}$ und
  - ein Wort w, auf dem $\mathcal{M}$ rechnen soll.
- $\mathcal{U}$ simuliert $\mathcal{M}$, indem sie jeweils nachschlägt, welchen δ-Übergang $\mathcal{M}$ machen würde.

Universelle Turing-Maschine

Turing-Maschine, die andere TMen simuliert

- Universelle TM $\mathcal{U}$ bekommt als Eingabe:
  - die Regelmenge einer beliebigen Turing-Maschine $\mathcal{M}$ und
  - ein Wort w, auf dem $\mathcal{M}$ rechnen soll.
- $\mathcal{U}$ simuliert $\mathcal{M}$, indem sie jeweils nachschlägt, welchen δ-Übergang $\mathcal{M}$ machen würde.

Universelle Turing-Maschine

TM als Eingabe für eine andere TM

Frage:

In welches Format fasst man die Regeln einer DTM $\mathcal{M}$ am besten, um sie einer universellen DTM als Eingabe zu geben?

Was muss man angeben, um eine DTM komplett zu beschreiben?

- das Alphabet,
- die Zustände,
- die δ-Übergänge
- den Startzustand.

Universelle Turing-Maschine

TM als Eingabe für eine andere TM

Frage:

In welches Format fasst man die Regeln einer DTM $\mathcal{M}$ am besten, um sie einer universellen DTM als Eingabe zu geben?

Was muss man angeben, um eine DTM komplett zu beschreiben?

- das Alphabet,
- die Zustände,
- die δ-Übergänge
- den Startzustand.

Universelle Turing-Maschine

TM als Eingabe für eine andere TM

Frage:

In welches Format fasst man die Regeln einer DTM $\mathcal{M}$ am besten, um sie einer universellen DTM als Eingabe zu geben?

Was muss man angeben, um eine DTM komplett zu beschreiben?

- das Alphabet,
- die Zustände,
- die δ-Übergänge
- den Startzustand.

Universelle Turing-Maschine

TM als Eingabe für eine andere TM

Frage:

In welches Format fasst man die Regeln einer DTM $\mathcal{M}$ am besten, um sie einer universellen DTM als Eingabe zu geben?

Was muss man angeben, um eine DTM komplett zu beschreiben?

- das Alphabet,
- die Zustände,
- die δ-Übergänge
- den Startzustand.

Universelle Turing-Maschine

TM als Eingabe für eine andere TM

Frage:
In welches Format fasst man die Regeln einer DTM $\mathcal{M}$ am besten, um sie einer universellen DTM als Eingabe zu geben?

Was muss man angeben, um eine DTM komplett zu beschreiben?

- das Alphabet,
- die Zustände,
- die δ-Übergänge
- den Startzustand.

Universelle Turing-Maschine

TM als Eingabe für eine andere TM

Frage:
In welches Format fasst man die Regeln einer DTM $\mathcal{M}$ am besten, um sie einer universellen DTM als Eingabe zu geben?

Was muss man angeben, um eine DTM komplett zu beschreiben?

- das Alphabet,
- die Zustände,
- die δ-Übergänge
- den Startzustand.

Universelle Turing-Maschine

Standardisierung von Alphabet, Zustandsmenge, Startzustand

- Unendliches Alphabet $\Sigma_\infty = \{a_0, a_1, \ldots\}$,
  so daß das Alphabet jeder DTM eine Teilmenge von Σ_∞ ist.
- Namen der Zustände einer DTM sind egal.
  Sie seien also $q_1, \ldots, q_n$
  (n kann dabei von DTM zu DTM verschieden sein).
- Sei q_1 immer der Startzustand, und
  bezeichne q_0 den Haltezustand

Damit:

**Wir können eine DTM komplett beschreiben,
indem wir nur ihre δ-Übergänge beschreiben.**

Universelle Turing-Maschine

Standardisierung von Alphabet, Zustandsmenge, Startzustand

- Unendliches Alphabet $\Sigma_\infty = \{a_0, a_1, \ldots\}$,
  so daß das Alphabet jeder DTM eine Teilmenge von Σ_∞ ist.
- Namen der Zustände einer DTM sind egal.
  Sie seien also $q_1, \ldots, q_n$
  (n kann dabei von DTM zu DTM verschieden sein).
- Sei q_1 immer der Startzustand, und
  bezeichne q_0 den Haltezustand

Damit:

**Wir können eine DTM komplett beschreiben,
indem wir nur ihre δ-Übergänge beschreiben.**

Universelle Turing-Maschine

Standardisierung von Alphabet, Zustandsmenge, Startzustand

- Unendliches Alphabet $\Sigma_\infty = \{a_0, a_1, \ldots\}$,
  so daß das Alphabet jeder DTM eine Teilmenge von Σ_∞ ist.
- Namen der Zustände einer DTM sind egal.
  Sie seien also $q_1, \ldots, q_n$
  (n kann dabei von DTM zu DTM verschieden sein).
- Sei q_1 immer der Startzustand, und
  bezeichne q_0 den Haltezustand

Damit:

**Wir können eine DTM komplett beschreiben,
indem wir nur ihre δ-Übergänge beschreiben.**

Universelle Turing-Maschine

Standardisierung von Alphabet, Zustandsmenge, Startzustand

- Unendliches Alphabet $\Sigma_\infty = \{a_0, a_1, \ldots\}$,
  so daß das Alphabet jeder DTM eine Teilmenge von Σ_∞ ist.
- Namen der Zustände einer DTM sind egal.
  Sie seien also $q_1, \ldots, q_n$
  (n kann dabei von DTM zu DTM verschieden sein).
- Sei q_1 immer der Startzustand, und
  bezeichne q_0 den Haltezustand

Damit:

**Wir können eine DTM komplett beschreiben,
indem wir nur ihre δ-Übergänge beschreiben.**

Universelle Turing-Maschine

Standardisierung von Alphabet, Zustandsmenge, Startzustand

- Unendliches Alphabet $\Sigma_\infty = \{a_0, a_1, \ldots\}$,
  so daß das Alphabet jeder DTM eine Teilmenge von Σ_∞ ist.
- Namen der Zustände einer DTM sind egal.
  Sie seien also $q_1, \ldots, q_n$
  (n kann dabei von DTM zu DTM verschieden sein).
- Sei q_1 immer der Startzustand, und
  bezeichne q_0 den Haltezustand

Damit:

**Wir können eine DTM komplett beschreiben,
indem wir nur ihre δ-Übergänge beschreiben.**

Universelle Turing-Maschine

(Mögliche) Kodierung der Übergangsrelation

Die DTM $\mathfrak{L}_{\#}$ habe die Regeln

$$q_1, \# \mapsto q_2, L \qquad q_2, \# \mapsto h, \#$$
$$q_1, | \mapsto q_2, L \qquad q_2, | \mapsto q_2, L$$

Dabei sei: $\# = a_0$ und $| = a_1$

Dann kann die DTM $\mathfrak{L}_{\#}$ so beschrieben werden:

	a_0	a_1
q_1	q_2, L	q_2, L
q_2	h, a_0	q_2, L

oder kürzer:

$$Z\ S\ 2\lambda\ S\ 2\lambda$$
$$Z\ S\ 00\ S\ 2\lambda$$

Universelle Turing-Maschine

(Mögliche) Kodierung der Übergangsrelation

Die DTM $\mathfrak{L}_{\#}$ habe die Regeln

$$q_1, \# \mapsto q_2, L \qquad q_2, \# \mapsto h, \#$$
$$q_1, | \mapsto q_2, L \qquad q_2, | \mapsto q_2, L$$

Dabei sei: $\# = a_0$ und $| = a_1$

Dann kann die DTM $\mathfrak{L}_{\#}$ so beschrieben werden:

	a_0	a_1
q_1	q_2, L	q_2, L
q_2	h, a_0	q_2, L

oder kürzer:

$$Z\ S\ 2\lambda\ S\ 2\lambda$$
$$Z\ S\ 00\ S\ 2\lambda$$

Universelle Turing-Maschine

(Mögliche) Kodierung der Übergangsrelation

Die DTM $\mathfrak{L}_{\#}$ habe die Regeln

$$q_1, \# \mapsto q_2, L \qquad q_2, \# \mapsto h, \#$$
$$q_1, | \mapsto q_2, L \qquad q_2, | \mapsto q_2, L$$

Dabei sei: $\# = a_0$ und $| = a_1$

Dann kann die DTM $\mathfrak{L}_{\#}$ so beschrieben werden:

	a_0	a_1
q_1	q_2, L	q_2, L
q_2	h, a_0	q_2, L

oder kürzer:

$$Z\ S\ 2\lambda\ S\ 2\lambda$$
$$Z\ S\ 00\ S\ 2\lambda$$

Universelle Turing-Maschine

(Mögliche) Kodierung der Übergangsrelation

Dabei steht:

- Z für “nächste Zeile”
- S für “nächste Spalte”
- λ für “links”, ρ für “rechts”
- die Zahl n für den n-ten Zustand und für das n-te Zeichen von Σ_∞.

Damit ist die DTM insgesamt durch ein einziges Wort beschrieben:

$$ZS2\lambda S2\lambda ZS00S2\lambda$$

Universelle Turing-Maschine

(Mögliche) Kodierung der Übergangsrelation

Dabei steht:

- Z für “nächste Zeile”
- S für “nächste Spalte”
- λ für “links”, ρ für “rechts”
- die Zahl n für den n-ten Zustand und für das n-te Zeichen von Σ_∞.

Damit ist die DTM insgesamt durch ein einziges Wort beschrieben:

$$ZS2\lambda S2\lambda ZS00S2\lambda$$

Universelle Turing-Maschine

Gödelisierung

Ein Verfahren, jeder Turing-Maschine eine Zahl oder ein Wort (**Gödelzahl** bzw. **Gödelwort**) so zuzuordnen, daß man aus der Zahl bzw. dem Wort die Turing-Maschine effektiv rekonstruieren kann.

Kurt Gödel

Kurt Gödel ⋆1906, †1978

- Bedeutendster Logiker des 20. Jahrhunderts
- Vollständigkeitssatz (1929)
  Promotion in Wien
- Unvollständigkeitssatz (1931)
  Idee er Gödilisierung
- Beweis der Unabhängigkeit der
  Kontinuuemshypothese
- Dozent in Princeton,
  befreundet mit Albert Einstein
- Tragischer Tod:
  Verfolgungswahn, Depressionen,
  Tod durch Unterernährung.

Kurt Gödel

Kurt Gödel ⋆1906, †1978

- Bedeutendster Logiker des 20. Jahrhunderts
- Vollständigkeitssatz (1929)
  Promotion in Wien
- Unvollständigkeitssatz (1931)
  Idee er Gödilisierung
- Beweis der Unabhängigkeit der
  Kontinuuemshypothese
- Dozent in Princeton,
  befreundet mit Albert Einstein
- Tragischer Tod:
  Verfolgungswahn, Depressionen,
  Tod durch Unterernährung.

Kurt Gödel

Kurt Gödel ⋆1906, †1978

- Bedeutendster Logiker des 20. Jahrhunderts
- Vollständigkeitssatz (1929)
  Promotion in Wien
- Unvollständigkeitssatz (1931)
  Idee er Gödilisierung
- Beweis der Unabhängigkeit der Kontinuuemshypothese
- Dozent in Princeton,
  befreundet mit Albert Einstein
- Tragischer Tod:
  Verfolgungswahn, Depressionen,
  Tod durch Unterernährung.

Kurt Gödel

Kurt Gödel ⋆1906, †1978

- Bedeutendster Logiker des 20. Jahrhunderts
- Vollständigkeitssatz (1929)
  Promotion in Wien
- Unvollständigkeitssatz (1931)
  Idee er Gödilisierung
- Beweis der Unabhängigkeit der Kontinuuemshypothese
- Dozent in Princeton,
  befreundet mit Albert Einstein
- Tragischer Tod:
  Verfolgungswahn, Depressionen,
  Tod durch Unterernährung.

Kurt Gödel

Kurt Gödel ⋆1906, †1978

- Bedeutendster Logiker des 20. Jahrhunderts
- Vollständigkeitssatz (1929)
  Promotion in Wien
- Unvollständigkeitssatz (1931)
  Idee er Gödilisierung
- Beweis der Unabhängigkeit der Kontinuuemshypothese
- Dozent in Princeton,
  befreundet mit Albert Einstein
- Tragischer Tod:
  Verfolgungswahn, Depressionen,
  Tod durch Unterernährung.

Kurt Gödel

Kurt Gödel ⋆1906, †1978

- Bedeutendster Logiker des 20. Jahrhunderts
- Vollständigkeitssatz (1929)
  Promotion in Wien
- Unvollständigkeitssatz (1931)
  Idee er Gödilisierung
- Beweis der Unabhängigkeit der Kontinuuemshypothese
- Dozent in Princeton,
  befreundet mit Albert Einstein
- Tragischer Tod:
  Verfolgungswahn, Depressionen,
  Tod durch Unterernährung.