GTI

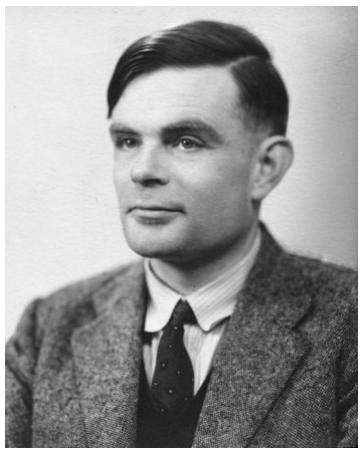
Hannes Diener

ENC B-0123,
diener@math.uni-siegen.de

6. Juni - 13. Juni

Die Turingmaschine war das erste (bzw. zweite) formale Modell der Berechenbarkeit. Sie wurden bereits 1936 (also lange bevor der erste moderne Computer überhaupt gebaut wurde) in dem Aufsatz „On computable numbers, with an application to the Entscheidungsproblem“ eingeführt. Viele der Ideen (universelle Turingmaschine, Halteproblem) die uns später begegnen finden sich bereits in dieser Arbeit!
Von praktischer Relevanz ist die Turingmaschine heute eher nicht. (Die von Neumann Architektur moderner Computer ist in vielen Punkten verschieden).

Etwas zum Namensgeber.



Alan Turing (1912–1954)
ist einer der zentralen Figuren in der Entwicklung des modernen Computers. Da viele seiner Leistungen während des zweiten Weltkrieges im Dienste von britischen Geheimdiensten erfolgten, waren sie lange nicht bekannt bzw. anerkannt. Ausserdem starb Turing jung und unter mysteriösen Umständen.

WARNUNG.

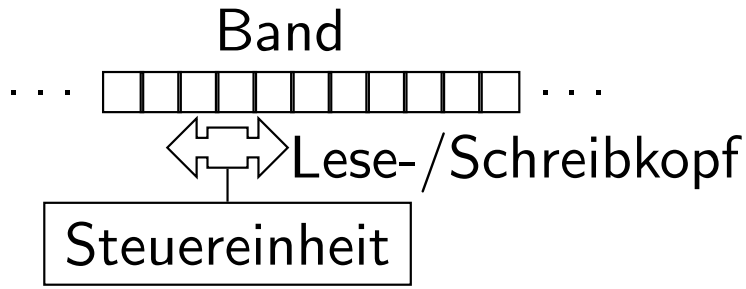
Es existieren eine Vielzahl von Varianten in der Definition von Turingmaschinen. Seien Sie also extrem kritisch, wenn Sie die Literatur, das Internet, oder ältere Kommilitonen benutzen um mehr über Turingmaschinen zu erfahren.

Ein bisschen halbnutzloses Wissen:
Turing hatte, als er die Turingmaschine definierte, kein Maschinenmodell im Hinterkopf sondern sogenannte „Computer“. Dies war ein regelrechter Beruf, in dem lange mathematische Berechnungen durchgeführt wurden (Anfang des 20. Jhd. mit einem hohen Frauenanteil). Die Analyse war, daß ein solcher Computer einen Stapel Papier hat (mit beliebig viel Nachschub) und jeweils auf das Papier Symbole schreiben kann und auch überschreiben kann (Bleistift und Radierer) je nachdem, was gerade auf dem Blatt steht, daß gerade behandelt wird.

Dies führte Turing zu folgendem Modell:

- Eine TM besteht aus einem unendlichen Band, daß in Zellen eingeteilt ist. In jeder Zelle steht ein Symbol aus einem endlichen Alphabet, oder die Zelle ist leer.
- Eine Steuereinheit befindet sich in einem von endlich vielen Zuständen.
- Ein Schreib-/Lesekopf befindet sich über einer der Zellen des Bandes.
- In Abhängigkeit des Zustandes und des Symbols der aktuellen Zelle wird vom Schreib-/Lesekopf das Symbol überschrieben und die Maschine bewegt sich entweder um eine Zelle nach links, bleibt stehen, oder bewegt sich um eine Zeile nach rechts.

Als Grafik:



Oder in LEGO

Definition

Eine (deterministische) Turingmaschine M ist eine Tupel

$$M = (Q, \Sigma, \Gamma, B, s, F, \delta)$$

bestehend aus

- *einer endlichen Menge von Zuständen Q*
- *einem Eingabealphabet Σ*
- *einem Bandalphabet Γ, so daß $\Sigma \subseteq \Gamma$*
- *ein Blanksymbol $B \in \Gamma \setminus \Sigma$*
- *einem Startzustand $s \in Q$*
- *einer Menge von Endzuständen $F \subseteq Q$*
- *einer partiellen Übergangsfunktion*

$$\delta : Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{L, R, N\}$$

Als Eingabe erhält die TM ein Wort auf dem Band, das ansonsten leer ist (also voll mit dem Blanksymbol). Sie startet am linken Ende des Wortes.

Wir lassen die Maschine solange laufen, bis sie in eine Endzustand wechselt.. Die Ausgabe ist das längste Wort am Lese-/Schreibkopf beginnend, daß kein Blanksymbol enthält.

Beispiel

Eine Maschine die zu einer Zahl in Binärdarstellung 1 *addiert.*

Beispiel

Eine Maschine die zu einer Zahl in Binärdarstellung 1 *addiert.*

Die Maschine $M_1 = (\{s, q, p, h\}, \{0, 1\}, \{0, 1, B\}, s, \{h\}, \delta)$ wobei δ gegeben ist durch

$$\delta(s, 0) = (s, 0, R) \qquad \delta(s, 1) = (s, 1, R)$$
$$\delta(s, B) = (q, B, L) \qquad \delta(q, 1) = (q, 0, L)$$
$$\delta(q, 0) = (p, 1, L) \qquad \delta(q, B) = (h, 1, N)$$
$$\delta(p, 0) = (p, 0, L) \qquad \delta(p, 1) = (p, 1, L)$$
$$\delta(p, B) = (h, B, R)$$

Die Idee zu M_1 ist, daß wir zuerst, im Zustand s ans rechte Ende des Wortes laufen und dann in den Zustand q wechseln. Ist die letzte Stelle eine 0, so überschreiben wir sie mit einer 1, ansonsten müssen wir, solange wir nur 1 finden diese durch 0 ersetzen, bis wir eine 0 finden, welche durch eine 1 ersetzt wird. Ebenfalls, falls das Wort nur aus 1 besteht, lesen wir das B links des Wortes und ersetzen dieses durch eine 1. Eventuell gehen wir dann in den Zustand p über und bewegen den Lese-/Schreibkopf ans linke Ende des Bands und gehen in den Endzustand h über.

Beispiel

Eine Maschine die überprüft ob die Eingabe von der Form $a^n b^n c^n$ ist.

Die Maschine $M_2 = \Big(\{s, q, p, h\}, \{a, b, c\}, \{a, \hat{a}, b, \hat{b}, c, \hat{c}, B\}, s, \{h\}, \delta\Big)$ wobei δ gegeben ist durch

$$\delta(s, a) = (p, \hat{a}, R)$$
$$\delta(p, x) = (p, x, R) \quad \text{für } x = a, \hat{b}$$
$$\delta(p, b) = (q, \hat{b}, R)$$
$$\delta(q, x) = (q, x, R) \quad \text{für } x = b, \hat{c}$$
$$\delta(q, c) = (m, \hat{c}, L)$$
$$\delta(m, x) = (m, x, R) \quad \text{für } x = a, b, \hat{b}, \hat{c}, c$$
$$\delta(m, \hat{a}) = (s, \hat{a}, R)$$
$$\delta(m, B) = (s, B, R)$$
$$\delta(s, x) = (s, x, R) \quad \text{für } x = \hat{b}, \hat{c}$$
$$\delta(s, B) = (h, B, N)$$

Die Idee ist, daß wir Zeichen markieren (x durch $\hat{x}$ ersetzen). Haben wir ein a markiert so suchen wir nach einem nicht markierten b und dann nach einem passenden c. Die Bedeutung der Zustände:

- p: Wir haben ein a gefunden und suchen ein passendes b.
- q: Wir haben ein passendes b gefunden und suchen ein nicht-markiertes c.
- m: Wir haben ein passendes c gefunden und suchen das linkste, nicht-markierte a.
- s: Der Startzustand, der aber auch gebraucht wird um, wenn kein unmarkiertes a gefunden wird, sicher zu stellen, daß es kein unmarkiertes b oder c gibt.
- h: Der Endzustand.

Ist die Eingabe von der Form $a^n b^n c^n$ so geht M_2 irgendwann in den Endzustand h über. Ansonsten bleibt sie stecken, d.h. liest einen Buchstaben z in einem Zustand y, so daß $\delta(y, z) = \perp$ ist.

Beispiel

Eine Maschine M_3 die ein Eingabewort w verdoppelt; also ww als Ausgabe hat.

Beispiel

Eine Maschine M_3 die ein Eingabewort w verdoppelt; also ww als Ausgabe hat.

Die Maschine hat drei Phasen:

1. $w \rightsquigarrow w\$$
2. $w\$ \rightsquigarrow w\$w$
3. $w\$w \rightsquigarrow ww$

wobei $\$$ ein Symbol ist, welches nicht im Eingabealphabet vorkommt. Details sind als Übung gelassen.

Wir wollen den Begriff der Berechnung jetzt noch weiter formalisieren (ähnlich wie bei endlichen und Kellerautomaten).

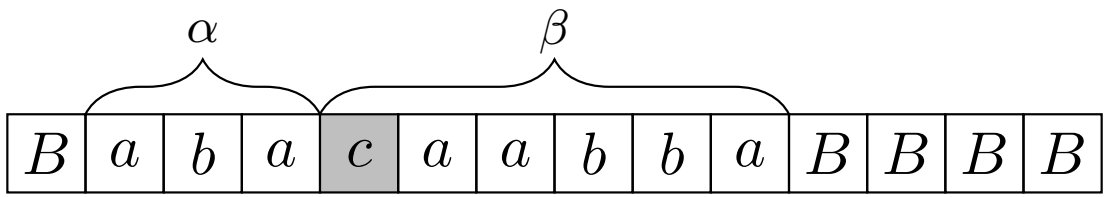
Definition

Eine Konfiguration einer Turingmaschine M beschreibt vollständig eine Momentaufnahme und ist ein Tripel

$$\kappa = (\alpha, q, \beta) \in \Gamma^* \times Q \times \Gamma^* \ .$$

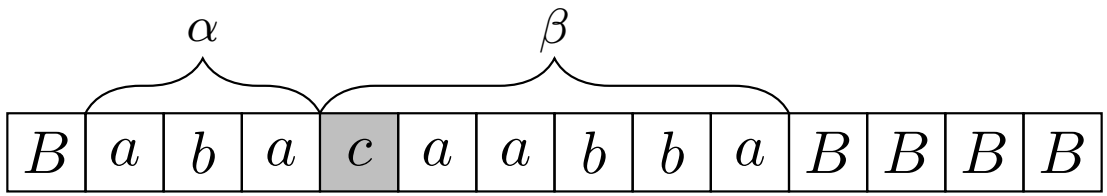
Wobei wir uns vorstellen, daß α das Wort links des Schreib-/Lesekopfs ist und β das rechts. Ausserdem ist q der Zustand der Turingmaschine.
Die Menge aller Konfiguration wollen wir mit $\mathrm{Conf}(M)$ bezeichnen.

Als Grafik:



Hierbei ist das graue Kästchen die Position des Lese-/Schreibkopfs.

Als Grafik:



Hierbei ist das graue Kästchen die Position des Lese-/Schreibkopfs.

(Man beachte, daß α und β das Blanksymbol enthalten können!)

Die Relation $\vdash_M$ soll, wie bei Automaten, einen einschrittigen Übergang zwischen zwei Konfigurationen erklären. Wir müssen hierbei eine Vielzahl von Fällen unterscheiden.

Definition

$\vdash_M \subseteq \mathrm{Conf}(M) \times \mathrm{Conf}(M)$ *ist definiert durch*

1. *Ist* $\delta(q, b_1) = (q', c, N)$, *so ist*

$$(\alpha, q, b_1 b_2 \dots b_n) \vdash_M (\alpha, q', c b_2 \dots b_n)$$

2. *Ist* $\delta(q, B) = (q', c, N)$

$$(\alpha, q, \varepsilon) \vdash_M (\alpha, q', c)$$

Definition

3. *Ist* $\delta(q, b_1) = (q', c, L)$, *so ist*

$$(a_1 \dots a_m, q, b_1\beta) \vdash_M (a_1 \dots a_{m-1}, q', a_m c\beta)$$

und

$$(\varepsilon, q, b_1\beta) \vdash_M (\varepsilon, q', Bc\beta)$$

4. *Ist* $\delta(q, B) = (q', c, L)$, *so ist*

$$(a_1 \dots a_m, q, \varepsilon) \vdash_M (a_1 \dots a_{m-1}, q', a_m c)$$

und

$$(\varepsilon, q, \varepsilon) \vdash_M (\varepsilon, q', Bc)$$

Definition

5. *Ist* $\delta(q, b_1) = (q', c, R)$, *so ist*

$$(\alpha, q, b_1 \dots b_n) \vdash_M (\alpha c, q', b_2 \dots b_n)$$

6. *Ist* $\delta(q, B) = (q', c, R)$, *so ist*

$$(\alpha, q, \varepsilon) \vdash_M (\alpha c, q', \varepsilon)$$

Wie immer bezeichnet $\vdash_M^*$ die reflexive, transitive Hülle der Relation $\vdash_M$. D.h. $\kappa \vdash_M^* \kappa'$ heißt wir kommen in einer endlichen Anzahl von Schritten von Konfiguration κ zur Konfiguration κ'.

Da δ partiell sein kann, kann es sein, daß es Konfigurationen κ gibt, von denen aus wir nicht weiter kommen, d.h. es gibt kein κ' so daß $\kappa \vdash_M = \kappa'$. Gibt es jedoch ein solches κ' ist es jedoch eindeutig.

Definition

Ist $w \in \Sigma^*$ ein Eingabewort, so ist die Konfiguration

$$\mathrm{init}_M(w) = (\varepsilon, s, w) \in \mathrm{Conf}(M)$$

die Startkonfiguration.

Beispiel

Für die Turingmaschine, die eine Binärzahl um eins erhöht erhalten wir formal:

$$\text{init}_M(1011) = (\varepsilon, s, 1011) \vdash \ldots$$

Sei $w \in \Sigma^*$. Die *Berechnung* von M für das Eingabewort w ist eine maximale (im Sinne von nicht verlängerbare) Folge

$$\text{init}_M(w) \vdash_M (\alpha_1, q_1, \beta_1) \vdash_M (\alpha_2, q_2, \beta_2) \vdash_M \ldots$$

von Konfigurationen. Es gibt drei Möglichkeiten:

1. Die Berechnung ist *unendlich*.
2. Die Berechnung ist *verwerfend*, d.h. endet in einer Konfiguration $(\alpha_n, q, b\beta_n)$ mit $\delta(q, b) = \bot$, bzw. einer Konfiguration $(\alpha_n, q, \varepsilon)$ mit $\delta(q, B) = \bot$.
3. Die Berechnung ist *akzeptierend*, d.h. endet (das erste mal) in einer Konfiguration (α_n, q_n, β_n) mit $q_n \in F$.

Mit ein klein wenig Arbeit können wir verwerfende Rechnungen ausschließen.

Satz

Ist M ein Turingmaschine, so können wir eine Maschine M' mit einer totalen Übergangsfunktion konstruieren, welche für ein Eingabewort w

- *mit der gleichen Konfiguration akzeptiert, mit der M akzeptiert*
- *die in einer unendlichen Berechnung resultiert, falls M dieses Verhalten hat*
- *in einer unendlichen Berechnung resultiert, falls M in einer verwerfenden Rechnung resultiert.*

Beweisidee:
Sei $M = (Q, \Sigma, \Gamma, B, s, F, \delta)$. Sei p ein Zustand, so daß $p \notin Q$. Sei nun δ' definiert durch

$$\delta'(q,x) = \begin{cases} \delta(q,x) & \text{falls } \delta(q,x) \text{ definiert ist} \\ (p,x,N) & \text{falls } \delta(q,x) = \perp \text{ oder } q = p\ . \end{cases}$$

D.h. im Fall daß $\delta(q,x)$ nicht definiert ist, wechseln wir in den Zustand p, der in einer künstlichen unendlichen Schleife steckenbleibt.

Man ist versucht das Ganze auch anderstherum zu machen: d.h. die unendlichen Rechnungen durch verwerfende zu ersetzen. Dies ist allerdings, im Allgemeinen, nicht möglich, wie wir später (Halteproblem) sehen werden.

Als nächstes wollen wir genau erklären, was es heißt für eine Funktion berechenbar zu sein.

Definition

Sei $M = (Q, \Sigma, \Gamma, B, s, F, \delta)$ eine Turingmaschine. Die von M berechnete Funktion $[\![M]\!] : \Sigma^* \rightharpoonup (\Gamma \setminus \{B\})^*$ ist definiert durch

- $[\![M]\!](w) = \perp$ *falls die Berechnung von M für w verwerfend oder unendlich ist*
- $[\![M]\!](w) = v$, *wobei die akzeptierende Berechnung von M für w in der Konfiguration (α_n, q_n, β_n) endet und v das längste Wort $\beta \in \Gamma \setminus \{B\}$, das Präfix von β ist.*

Beispiel

Für M_1 *gilt, daß* $[\![M_1]\!](1011) = 1100$

Definition

Seien Σ und Λ Alphabete. Eine Funktion $f : \Sigma^* \rightharpoonup \Lambda^*$ heißt Turing-berechenbar, falls es eine Turingmaschine M gibt, so daß

$$[\![M]\!] = f \ .$$

D.h. $[\![M]\!]$ ist definiert genau dann, wenn f definiert ist und in diesem Fall stimmen die Werte überein.
Man beachte auch, daß Γ eine echte Obermenge von Λ sein kann.

Wie am Anfang des Themas gesagt interessieren wir uns ja vor Allem an der Berechenbarkeit von partiellen Funktionen zwischen natürlichen Zahlen. Kodieren wir diese als Wörter können wir den bereits definierten Begriff verwenden.

Satz

Bekanntermaßen (DMI) gilt: zu jeder natürlichen Zahl n gibt es eine eindeutige Binärdarstellung, das ist ein Wort $b_m \dots b_1 b_0 \in \{0,1\}^*$ so daß

- *für $m > 0$ gilt $b_m = 1$*
- $n = \sum_{i=0}^{m} b_i 2^i$.

Die Funktion $\mathrm{bin} : \mathbb{N} \to \{0,1\}^*$ ist die Funktion, die jeder natürlichen Zahl diese Darstellung zuordnet.

Definition

Eine partielle Funktion $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ *heißt Turing-berechenbar, falls die Funktion* $\hat{f} : \{0, 1, \#\}^* \to \{0, 1\}^*$ *definiert durch*

$$\hat{f}(\mathrm{bin}(n_1)\# \ldots \#\, \mathrm{bin}(n_k)) = \mathrm{bin}(f(n_1, \ldots, n_k))$$

und $\hat{f} = \perp$ *für alle anderen Eingaben oder falls* $f(n_1, \ldots, n_k) = \perp$*, Turing-berechenbar ist.*

Beispiel

Das die (Nachfolger-) Funktion $S : \mathbb{N} \to \mathbb{N}$ definiert durch $S(n) = n + 1$ Turing-berechenbar ist folgt aus dem Beispiel M_1.

Beobachtung

Wir können mit einer Turingmaschine überprüfen ob ein Eingabewort von der Form

$$\mathrm{bin}(n_1)\# \ldots \#\, \mathrm{bin}(n_k)$$

ist oder nicht. Genauer es gibt eine Turingmaschine $T_{bin?}$ welche bei Eingabe der Form $\mathrm{bin}(n_1)\# \ldots \#\, \mathrm{bin}(n_k)$ in einer akzeptierenden Konfiguration $(\varepsilon, q, \mathrm{bin}(n_1)\# \ldots \#\, \mathrm{bin}(n_k))$ endet und sonst verwirft.

Wie schon informell bei Beispiel M_3 geschehen können wir Turingmaschinen hintereinanderschalten. Diese Idee wollen wir etwas formalisieren. Sei also T_1, T_2 Turingmaschinen. Die Maschine $T_1; T_2$ ist die Turingmaschine, die sich zunächst wie T_1 verhält und sobald T_1 akzeptiert in den Startzustand von T_2 wechselt.

Formal: Sei $T_i = (Q_i, \Sigma, \Gamma, B, s_i, F_i, \delta_i)$ für $i = 1, 2$. Wir nehmen an, daß $Q_1 \cap Q_2 = \emptyset$. Dann ist

$$T_1; T_2 = (Q_1 \cup Q_2, \Sigma, \Gamma, B, s_1, F_2, \delta) \ ,$$

wobei δ definiert ist durch

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{falls } q \in Q_1 \setminus F_1 \\ \delta_2(q,a) & \text{falls } q \in Q_2 \\ (s_2, a, N) & \text{falls } q \in F_1 \ . \end{cases}$$

Satz

Seien T_1, T_2 wie oben. Des Weiteren nehmen wir an, daß jede akzeptierende Berechnung von T_1 in einer Konfiguration der Form $(\alpha, q, a_1 a_2 \ldots a_n \beta)$ endet, wobei $\alpha, \beta \in B^*$ und $a_1, \ldots, a_n \in \Sigma$. Dann gilt

$$[\![T_1; T_2]\!] = [\![T_2]\!] \circ [\![T_1]\!] \ .$$

Die folgenden Funktionen sind Turing-berechenbar

1. $\mathrm{id} : \mathbb{N} \to \mathbb{N}$
2. $\pi_j^{(k)} : \mathbb{N}^k \to \mathbb{N}$ definiert durch $\pi_j^{(k)}(n_1, \ldots, n_k) = n_j$ (die Projektion auf die j-te Koordinate)
3. $f_- : \mathbb{N} \to \mathbb{N}$ definiert durch $f(n) = \max\{0, n-1\}$
4. $f_{\mathrm{add}} : \mathbb{N}^2 \to \mathbb{N}$ definiert durch $f(n,m) = n + m$
5. $f_{\mathrm{sub}} : \mathbb{N}^2 \to \mathbb{N}$ definiert durch $f(n,m) = \max\{0, n-m\}$
6. ...

Die Konstruktion dieser Maschinen ist nicht schwer, man beachte jedoch, daß man zuerst die Maschine $T_{bin?}$ simulieren sollte. Die Funktionen f_{add} und f_{sub} lassen sich am Besten mit Mehrbandmaschinen berechnen. Dazu gleich mehr.

Varianten der Turingmaschine.

- Mehrband
- beschränktes Alphabet
- nicht-deterministisch

Eine zunächst mächtiger erscheinende Erweiterung der Turingmaschine ist die *Mehrband-Turingmaschine*. Die Vorstellung ist, daß wir nun mehrere Bänder haben, mit unabhängig voneinander bewegbaren Lese-/Schreibköpfen.

Definition

Eine (deterministische) k-Band-Turingmaschine M ist eine Tupel

$$M = (Q, \Sigma, \Gamma, B, s, F, \delta)$$

bestehend aus den üblichen Verdächtigen jedoch mit

- *einer partiellen Übergangsfunktion*

$$\delta : Q \times \Gamma^k \rightharpoonup Q \times (\Gamma \times \{L, R, N\})^k \ .$$

(Im Falle $k = 1$ haben wir also genau unsere normale Turingmaschine.)

Analog zur Einbandmaschine ist eine Konfiguration einer k-Band-Turingmaschine ein Tripel

$$(\langle \alpha_1, \ldots, \alpha_k \rangle, q, \langle \beta_1, \ldots, \beta k \rangle)$$

welche beschreibt, daß die Maschine im Zustand q ist und auf dem i-ten Band α_i links des Lese-/Schreibkopfs steht und β_i rechts davon.

Die Übergangsrelation $\vdash$ ist analog zur Einbandmaschine definiert.

Als Eingabe und Ausgabeband dient das erste; alle anderen Bänder sind leer. Der Begriff der k-Band-Turing-Berechenbarkeit ist ebenfalls analog definiert.

Viele Probleme lassen sich leichter mit Mehrbandmaschinen lösen. Das Verdoppeln von Wörtern durch M_3, lässt sich mit einer 2-Band Turingmaschine lösen, die zunächst das Eingabewort w auf das zweite Band kopiert, dann den Lese-/Schreibkopf des zweiten Bandes zurück zum Anfang des Wortes bewegt und dann dieses wieder auf das erste Band rechts des originalen Wortes kopiert.

Formal: $M_3' = (\{s, q, p, m, h\}, \Sigma, \Sigma \cup \{B\}, B, s, \{h\}, \delta)$ wobei δ definiert ist durch

$$\delta(s, (x, B)) = (s, (x, R), (x, R)) \quad \text{für } x \in \Sigma$$
$$\delta(s, (B, B)) = (q, (B, N), (B, L))$$
$$\delta(q, (B, x)) = (q, (B, N), (x, L)) \quad \text{für } x \in \Sigma$$
$$\delta(q, (B, B)) = (p, (B, N), (B, R))$$
$$\delta(p, (B, x)) = (p, (x, R), (x, R)) \quad \text{für } x \in \Sigma$$
$$\delta(p, (B, B)) = (m, (B, L), (B, N))$$
$$\delta(m, (x, B)) = (m, (x, L), (B, N)) \quad \text{für } x \in \Sigma$$
$$\delta(m, (B, B)) = (h, (B, R), (B, N))$$

Die Maschine M_3' benötigt außerdem weniger Zustände, kein zusätzliches Symbol und viel weniger Schritte um die Rechnung auszuführen als M_3!

Überraschenderweise gilt jedoch:

Satz

Eine Funktion $f : \Sigma^* \to \Lambda^*$ ist genau dann k-Band-Turing-berechenbar, wenn sie (1-Band-) Turing-berechenbar ist.

Überraschenderweise gilt jedoch:

Satz

Eine Funktion $f : \Sigma^* \to \Lambda^*$ ist genau dann k-Band-Turing-berechenbar, wenn sie (1-Band-) Turing-berechenbar ist.

Beweisidee: Vorlesung.

Überraschenderweise gilt jedoch:

Satz

Eine Funktion $f : \Sigma^* \to \Lambda^*$ ist genau dann k-Band-Turing-berechenbar, wenn sie (1-Band-) Turing-berechenbar ist.

Beweisidee: Vorlesung.

Man beachte jedoch, daß diese Simulation sehr viel Zeit kosten kann!

Beschränktes Alphabet:
Sei Σ ein Alphabet mit höchstens 2^n Symbolen. Jedes dieser Symbole lässt sich durch einen Binärcode der exakten Länge n kodieren. Sagen wir z.B. $\Sigma = \{a, b, c, d\}$, dann können wir folgende Kodierung verwenden: $a = 00$, $b = 01$, $c = 10$, und $d = 11$. Für $x \in \Sigma$, bezeichne $\tilde{x}$ eine entsprechende Kodierung. Diese können wir natürlich auch auf Wörter über Σ ausdehnen:

$$\widetilde{a_1 \dots a_n} = \tilde{a_1} \dots \tilde{a_n}$$

In obigen Beispiel gilt also

$$\widetilde{aba} = 000100 \ .$$

Es gilt.

Satz

Sei $M = (Q, \Sigma, \Gamma, B, s, F, \delta)$ eine Turingmaschine. Dann gibt es eine Turingmaschine $M' = (Q', \{0,1\}, \{0,1,B\}, B, s', F', \delta')$, so daß für alle $w \in \Sigma^*$ M bei Eingabe w genau dann akzeptiert wenn M' bei Eingabe $\tilde{w}$ akzeptiert und in diesem Fall gilt

$$\widetilde{[\![M]\!](w)} = [\![M']\!](\tilde{w}) \ .$$

D.h. wir können annehmen, daß eine Turingmaschine als Eingabealphabet nur $\{0,1\}$ und als Bandalphabet nur $\{0,1,B\}$ hat!

Analog des Unterschieds zwischen DEAs und NDEAs kann man auch bei Turingmaschinen eine Übergangsrelation Δ an Stelle der Übergangsfunktion δ benutzen. Fast alle anderen Definitionen (Konfiguration, ...) übertragen sich analog.

Analog des Unterschieds zwischen DEAs und NDEAs kann man auch bei Turingmaschinen eine Übergangsrelation Δ an Stelle der Übergangsfunktion δ benutzen. Fast alle anderen Definitionen (Konfiguration, ...) übertragen sich analog.

Es kann aber jetzt vorkommen, daß es zu einer Konfiguration κ eventuell zwei verschieden Konfigurationen κ' und κ'' gibt, so daß

$$\kappa \vdash \kappa' \quad \text{und} \quad \kappa \vdash \kappa'' \ .$$

Es kann also viele verschieden Rechnungen geben. Alle möglichen Rechnungen kann man sich als Baum vorstellen.

Analog des Unterschieds zwischen DEAs und NDEAs kann man auch bei Turingmaschinen eine Übergangsrelation Δ an Stelle der Übergangsfunktion δ benutzen. Fast alle anderen Definitionen (Konfiguration, ...) übertragen sich analog.

Es kann aber jetzt vorkommen, daß es zu einer Konfiguration κ eventuell zwei verschieden Konfigurationen κ' und κ'' gibt, so daß

$$\kappa \vdash \kappa' \quad \text{und} \quad \kappa \vdash \kappa'' \ .$$

Es kann also viele verschieden Rechnungen geben. Alle möglichen Rechnungen kann man sich als Baum vorstellen.

Damit wir immer noch auf sinnvolle Art und Weise über die von nicht-deterministischen Turingmaschinen berechnete Funktion reden können, machen wir noch folgende Einschränkungen: Wir fordern zusätzlich, daß jede akzeptierende Rechnung mit dem gleichen Resultat endet. Eine nicht-deterministische Turingmaschine akzeptiert ein Eingabewort, wenn es mindestens eine akzeptierende Rechnung für dieses Eingabewort gibt.

Auch hier gibt es wieder Probleme, die sich leichter mit einer nicht-deterministischen als mit einer normalen Turingmaschine lösen lassen.

Überraschenderweise gilt auch hier:

Satz

Eine Funktion $f : \Sigma^* \to \Lambda^*$ ist genau dann von einer nicht-deterministischen Turingmaschine berechenbar, wenn sie Turing-berechenbar ist.

Beweis.

Technisch. Idee in der Vorlesung, Details in der Literatur. □

Man beachte jedoch, daß auch diese Simulation extrem viel Zeit kosten kann!