Script generated by TTT

Title: seidl: Theoretische_Informatik (04.06.2012)

Date: Mon Jun 04 10:15:29 CEST 2012

Duration: 89:08 min

Pages: 51

Beispiel 4.8 (Unär +1)

$$M = (\{q, f\}, \{1\}, \{1, \square\}, \delta, q, \square, \{f\})$$

Beispiel 4.8 (Unär +1)

$$M = (\{q, f\}, \{1\}, \{1, \square\}, \delta, q, \square, \{f\})$$

$$\delta(q, \square) = (f, 1, N)$$

Beispiel 4.8 (Unär +1)

$$M = (\{q, f\}, \{1\}, \{1, \square\}, \delta, q, \square, \{f\})$$

$$\begin{aligned} \delta(q, \square) &= (f, 1, N) \\ \delta(q, 1) &= (q, 1, R) \end{aligned}$$

Beispiel 4.8 (Unär +1)

$$M = (\{q, f\}, \{1\}, \{1, \Box\}, \delta, q, \Box, \{f\})$$

$$\begin{aligned} \delta(q, \Box) &= (f, 1, N) \\ \delta(q, 1) &= (q, 1, R) \end{aligned}$$

- Beispiellauf:

$(\epsilon, q, 11) \to_M$

Beispiel 4.8 (Unär +1)

$$M = (\{q, f\}, \{1\}, \{1, \Box\}, \delta, q, \Box, \{f\})$$

$$\begin{aligned} \delta(q, \Box) &= (f, 1, N) \\ \delta(q, 1) &= (q, 1, R) \end{aligned}$$

- Beispiellauf:

$(\epsilon, q, 11) \to_M$

Beispiel 4.8 (Unär +1)

$$M = (\{q, f\}, \{1\}, \{1, \Box\}, \delta, q, \Box, \{f\})$$

$$\begin{aligned} \delta(q, \Box) &= (f, 1, N) \\ \delta(q, 1) &= (q, 1, R) \end{aligned}$$

- Beispiellauf:

$(\epsilon, q, 11) \to_M (1, q, 1) \to_M$

Beispiel 4.8 (Unär +1)

$$M = (\{q, f\}, \{1\}, \{1, \Box\}, \delta, q, \Box, \{f\})$$

$$\begin{aligned} \delta(q, \Box) &= (f, 1, N) \\ \delta(q, 1) &= (q, 1, R) \end{aligned}$$

- Beispiellauf:

$(\epsilon, q, 11) \to_M (1, q, 1) \to_M (11, q, \epsilon) \to_M$

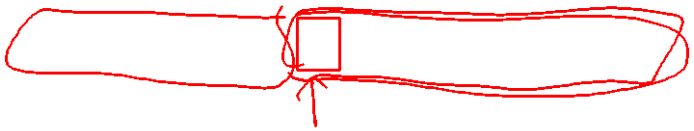
Beispiel 4.8 (Unär +1)

$$M = (\{q, f\}, \{1\}, \{1, \square\}, \delta, q, \square, \{f\})$$

$$\begin{aligned} \delta(q, \square) &= (f, 1, N) \\ \delta(q, 1) &= (q, 1, R) \end{aligned}$$

- Beispiellauf:

$$(\epsilon, q, 11) \to_M (1, q, 1) \to_M (11, q, \epsilon) \to_M (11, f, 1)$$



Beispiel 4.8 (Unär +1)

$$M = (\{q, f\}, \{1\}, \{1, \square\}, \delta, q, \square, \{f\})$$

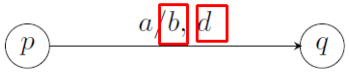
$$\begin{aligned} \delta(q, \square) &= (f, 1, N) \\ \delta(q, 1) &= (q, 1, R) \end{aligned}$$

- Beispiellauf:

$$(\epsilon, q, 11) \to_M (1, q, 1) \to_M (11, q, \epsilon) \to_M (11, f, 1)$$

- Welche Eingaben führen von q nach f?

Eine graphische Notation für $\delta(p, a) = (q, b, d)$:



Beispiel 4.9 (Binär +1)

ZB $1011 \mapsto 1100$

Beispiel 4.9 (Binär +1)

ZB $1011 \mapsto 1100$

$$M = (\{q_0, q_1, q_2, q_f\}, \{0,1\}, \{0,1,\Box\}, \delta, q_0, \Box, \{q_f\})$$

wobei

$$\begin{array}{lll} \delta(q_0,0) = (q_0,0,R) & \delta(q_1,1) = (q_1,0,L) & \delta(q_2,0) = (q_2,0,L) \\ \delta(q_0,1) = (q_0,1,R) & \delta(q_1,0) = (q_2,1,L) & \delta(q_2,1) = (q_2,1,L) \\ \delta(q_0,\Box) = (q_1,\Box,L) & \delta(q_1,\Box) = (q_f,1,N) & \delta(q_2,\Box) = (q_f,\Box,R) \end{array}$$

Beispiel 4.9 (Binär +1)

ZB $1011 \mapsto 1100$

$$M = (\{q_0, q_1, q_2, q_f\}, \{0,1\}, \{0,1,\Box\}, \delta, q_0, \Box, \{q_f\})$$

wobei

$$\begin{array}{lll} \delta(q_0,0) = (q_0,0,R) & \delta(q_1,1) = (q_1,0,L) & \delta(q_2,0) = (q_2,0,L) \\ \delta(q_0,1) = (q_0,1,R) & \delta(q_1,0) = (q_2,1,L) & \delta(q_2,1) = (q_2,1,L) \\ \delta(q_0,\Box) = (q_1,\Box,L) & \delta(q_1,\Box) = (q_f,1,N) & \delta(q_2,\Box) = (q_f,\Box,R) \end{array}$$

Beispiellauf:

$$\begin{array}{l} (\epsilon, q_0, 101) \to_M (1, q_0, 01) \to_M (10, q_0, 1) \to_M (101, q_0, \epsilon) \to_M \\ (10, q_1, 1\Box) \to_M (1, q_1, 00\Box) \to_M \\ (\epsilon, q_2, 110\Box) \to_M (\epsilon, q_2, \Box 110\Box) \to_M \\ (\Box, q_f, 110\Box) \end{array}$$

Definition 4.10

Eine Turingmaschine M akzeptiert die Sprache

$$L(M) = \{w \in \Sigma^* \mid \exists q \in F, \alpha, \beta \in \Gamma^*.\ (\epsilon, q_0, w) \to_M^* (\alpha, q, \beta)\}$$

Definition 4.10

Eine Turingmaschine M akzeptiert die Sprache

$$L(M) = \{w \in \Sigma^* \mid \exists q \in F, \alpha, \beta \in \Gamma^*.\ (\epsilon, q_0, w) \to_M^* (\alpha, q, \beta)\}$$

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt Turing-berechenbar gdw es eine Turingmaschine M gibt, so dass für alle $n_1, \ldots n_k, m \in \mathbb{N}$ gilt

$$\begin{array}{l} f(n_1, \ldots, n_k) = m \quad \Leftrightarrow \\ \exists r \in F.\ (\epsilon, q_0, bin(n_1)\#bin(n_2)\#\ldots\#bin(n_k)) \\ \qquad \to_M^* (\Box\ldots\Box, r, bin(m)\Box\ldots\Box) \end{array}$$

wobei $bin(n)$ die Binärdarstellung der Zahl n ist.

Definition 4.10

Eine Turingmaschine M akzeptiert die Sprache

$$L(M) = \{w \in \Sigma^* \mid \exists q \in F, \alpha, \beta \in \Gamma^*.\ (\epsilon, q_0, w) \to_M^* (\alpha, q, \beta)\}$$

Eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt Turing-berechenbar gdw es eine Turingmaschine M gibt, so dass für alle $n_1, \ldots n_k, m \in \mathbb{N}$ gilt

$$\begin{aligned} & f(n_1, \ldots, n_k) = m \quad \Leftrightarrow \\ & \exists r \in F.\ (\epsilon, q_0, bin(n_1)\#bin(n_2)\#\ldots\#bin(n_k)) \\ & \qquad \to_M^* (\square\ldots\square, r, bin(m)\square\ldots\square) \end{aligned}$$

wobei $bin(n)$ die Binärdarstellung der Zahl n ist.

Eine Funktion $f : \Sigma^* \to \Sigma^*$ heißt Turing-berechenbar gdw es eine Turingmaschine M gibt, so dass für alle $u, v \in \Sigma^*$ gilt

$$f(u) = v \quad \Leftrightarrow \quad \exists r \in F.\ (\epsilon, q_0, u) \to_M^* (\square\ldots\square, r, v\square\ldots\square)$$

Zum Halten/Terminieren von TM

OE(!) können wir festlegen, dass eine TM in der Konfiguration (α, q, β) hält falls $q \in F$.

Zum Halten/Terminieren von TM

OE(!) können wir festlegen, dass eine TM in der Konfiguration (α, q, β) hält falls $q \in F$. Dann gibt es keine Konfiguration (α', q', β') mit $(\alpha, q, \beta) \to_M (\alpha', q', \beta')$.

Ist δ partiell, so kann eine TM auch halten, bevor sie einen Endzustand erreicht.

Daher können wir auch annehmen, dass $\delta(q, a)$ undefiniert (bzw $\emptyset$) ist falls $q \in F$.

Satz 4.11

Zu jeder nichtdeterministischen TM N gibt es eine deterministische TM M mit $L(N) = L(M)$.

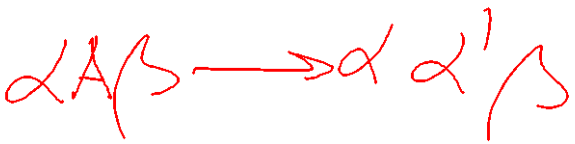
Satz 4.11
Zu jeder nichtdeterministischen TM N gibt es eine deterministische TM M mit $L(N) = L(M)$.

Beweis:
M durchsucht den Baum der Berechnungen von N in *Breitensuche*, beginnend mit der Startkonfiguration (ϵ, q_0, w), eine Ebene nach der anderen.

Satz 4.11
Zu jeder nichtdeterministischen TM N gibt es eine deterministische TM M mit $L(N) = L(M)$.

Beweis:
M durchsucht den Baum der Berechnungen von N in *Breitensuche*, beginnend mit der Startkonfiguration (ϵ, q_0, w), eine Ebene nach der anderen.
Gibt es in dem Baum (auf Ebene n) eine Konfigurationen mit Endzustand, so wird diese (nach Zeit $O(c^n)$) gefunden. □

Satz 4.12
Die von Turingmaschinen akzeptierten Sprachen sind genau die Typ-0-Sprachen der Chomsky Hierarchie.



Satz 4.12
Die von Turingmaschinen akzeptierten Sprachen sind genau die Typ-0-Sprachen der Chomsky Hierarchie.

Beweis:
Wir beschreiben nur die Beweisidee. (Mehr Details: [Schöning]).

„$\Longrightarrow$“: Grammatikregeln können direkt die Rechenregeln einer TM simulieren.

Satz 4.12
Die von Turingmaschinen akzeptierten Sprachen sind genau die Typ-0-Sprachen der Chomsky Hierarchie.

Beweis:
Wir beschreiben nur die Beweisidee. (Mehr Details: [Schöning]).

„$\Longrightarrow$“: Grammatikregeln können direkt die Rechenregeln einer TM simulieren.

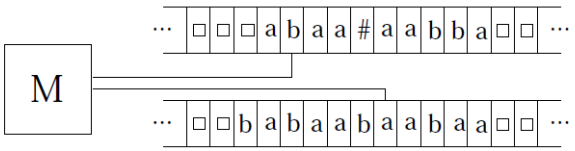
Satz 4.12
Die von Turingmaschinen akzeptierten Sprachen sind genau die Typ-0-Sprachen der Chomsky Hierarchie.

Beweis:
Wir beschreiben nur die Beweisidee. (Mehr Details: [Schöning]).

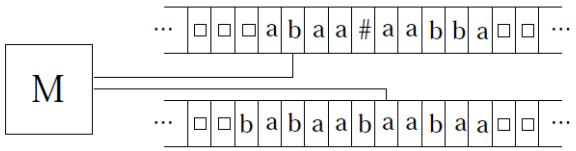
„$\Longrightarrow$“: Grammatikregeln können direkt die Rechenregeln einer TM simulieren.

„$\Longleftarrow$“: Die (nichtdeterministische) TM versucht von ihrer Eingabe aus das Startsymbol der Grammatik zu erreichen, indem sie die Produktionen der Grammatik von rechts nach links anwendet, (nichtdeterministisch) an jeder möglichen Stelle. □

Eine beliebte Modellvariante ist die k-Band-Turingmaschine:



Eine beliebte Modellvariante ist die k-Band-Turingmaschine:



Die k Köpfe sind völlig unabhängig voneinander:

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, N\}^k$$

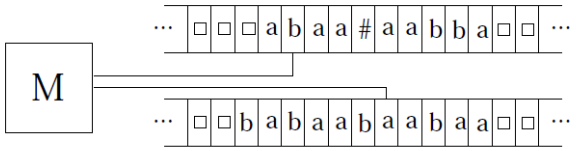
Satz 4.13

Jede k-Band-Turingmaschine kann effektiv durch eine 1-Band-TM simuliert werden.

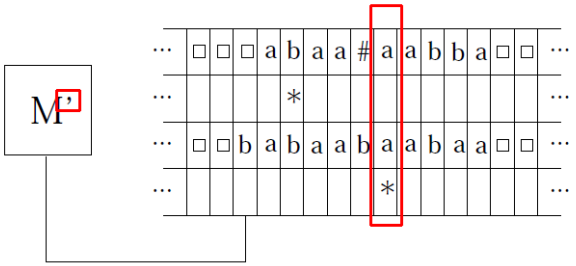
Satz 4.13

Jede k-Band-Turingmaschine kann effektiv durch eine 1-Band-TM simuliert werden.

Beweisidee: Aus



wird



Beweisskizze:

- $\Gamma' := (\Gamma \times \{\star, \square\})^k$
- M' simuliert einen M-Schritt durch mehrere Schritte: M'
  - startet mit Kopf links von allen $\star$.

Beweisskizze:

- $\Gamma' := (\Gamma \times \{\star, \square\})^k$
- M' simuliert einen M-Schritt durch mehrere Schritte: M'
  - startet mit Kopf links von allen $\star$.
  - geht nach rechts bis alle $\star$ überschritten sind, und merkt sich dabei (in Q') die Zeichen über jedem $\star$.
  - hat jetzt alle Information, um δ_M anzuwenden.

Beweisskizze:

- $\Gamma' := (\Gamma \times \{\star, \square\})^k$
- M' simuliert einen M-Schritt durch mehrere Schritte: M'
  - startet mit Kopf links von allen $\star$.
  - geht nach rechts bis alle $\star$ überschritten sind, und merkt sich dabei (in Q') die Zeichen über jedem $\star$.
  - hat jetzt alle Information, um δ_M anzuwenden.
  - geht nach links über alle $\star$ hinweg und führt dabei δ_M aus.

Beweisskizze:

- $\Gamma' := (\Gamma \times \{\star, \square\})^k$
- M' simuliert einen M-Schritt durch mehrere Schritte: M'
  - startet mit Kopf links von allen $\star$.
  - geht nach rechts bis alle $\star$ überschritten sind, und merkt sich dabei (in Q') die Zeichen über jedem $\star$.
  - hat jetzt alle Information, um δ_M anzuwenden.
  - geht nach links über alle $\star$ hinweg und führt dabei δ_M aus.

Beobachtung:

n Schritte von M lassens sich durch
$O(n^2)$ Schritte von M' simulieren.

Beweisskizze:

- $\Gamma' := (\Gamma \times \{\star, \square\})^k$
- M' simuliert einen M-Schritt durch mehrere Schritte: M'
  - startet mit Kopf links von allen $\star$.
  - geht nach rechts bis alle $\star$ überschritten sind, und merkt sich dabei (in Q') die Zeichen über jedem $\star$.
  - hat jetzt alle Information, um δ_M anzuwenden.
  - geht nach links über alle $\star$ hinweg und führt dabei δ_M aus.

Beobachtung:

n Schritte von M lassens sich durch
$O(n^2)$ Schritte von M' simulieren.

Denn nach n Schritten von M trennen $\leq 2n$ Felder linkesten und rechtesten Kopf.

4.3 Programmieren von Mehrband-TM

Die folgenden Basismaschinen sind leicht programmierbar:

- Band i := Band $i + 1$
- Band i := Band $i - 1$
- Band i := 0
- Band i := Band j

Seien $M_i = (Q_i, \Sigma, \Gamma_i, \delta_i, q_i, \square, F_i)$, $i = 1, 2$.
Die sequentielle Komposition (Hintereinanderschaltung) von M_1 und M_2 bezeichnen wir mit

$$\longrightarrow M_1 \longrightarrow M_2 \longrightarrow$$

$P_1 ; P_2$

Seien $M_i = (Q_i, \Sigma, \Gamma_i, \delta_i, q_i, \square, F_i)$, $i = 1, 2$.
Die sequentielle Komposition (Hintereinanderschaltung) von M_1 und M_2 bezeichnen wir mit

$$\longrightarrow M_1 \longrightarrow M_2 \longrightarrow$$

Sie ist wie folgt definiert:

$$M := (Q_1 \cup Q_2, \Sigma, \Gamma_1 \cup \Gamma_2, \delta, q_1, \square, F_2)$$

wobei (oE) $Q_1 \cap Q_2 = \emptyset$

Seien $M_i = (Q_i, \Sigma, \Gamma_i, \delta_i, q_i, \square, F_i)$, $i = 1, 2$.
Die sequentielle Komposition (Hintereinanderschaltung) von M_1 und M_2 bezeichnen wir mit

$$\longrightarrow M_1 \longrightarrow M_2 \longrightarrow$$

Sie ist wie folgt definiert:

$$M := (Q_1 \cup Q_2, \Sigma, \Gamma_1 \cup \Gamma_2, \delta, q_1, \square, F_2)$$

wobei (oE) $Q_1 \cap Q_2 = \emptyset$ und

$$\delta := \delta_1 \cup \delta_2 \cup \{(f_1, a) \mapsto (q_2, a, N) \mid f_1 \in F_1, a \in \Gamma_1\}$$

Sind f_1 und f_2 Endzustände von M so bezeichnet

$$\begin{array}{ccccc} \longrightarrow & M & \xrightarrow{f_1} & M_1 & \longrightarrow \\ & \downarrow f_2 & & & \\ & M_2 & & & \\ & \downarrow & & & \end{array}$$

eine Fallunterscheidung,

Sind f_1 und f_2 Endzustände von M so bezeichnet

$$\begin{array}{lcl} \longrightarrow & M \xrightarrow{f_1} & M_1 \longrightarrow \\ & \downarrow f_2 & \\ & M_2 & \\ & \downarrow & \end{array}$$

eine Fallunterscheidung, dh eine TM, die vom Endzustand f_1 von M nach M_1 übergeht, und von f_2 aus nach M_2.

Die folgende TM nennen wir „Band=0?“ (bzw „Band $i = 0$?“):

Sind f_1 und f_2 Endzustände von M so bezeichnet

$$\begin{array}{lcl} \longrightarrow & M \xrightarrow{f_1} & M_1 \longrightarrow \\ & \downarrow f_2 & \\ & M_2 & \\ & \downarrow & \end{array}$$

eine Fallunterscheidung, dh eine TM, die vom Endzustand f_1 von M nach M_1 übergeht, und von f_2 aus nach M_2.

Die folgende TM nennen wir „Band=0?“ (bzw „Band $i = 0$?“):

$$\delta(q_0, 0) = (q_0, 0, R)$$

Sind f_1 und f_2 Endzustände von M so bezeichnet

$$\begin{array}{lcl} \longrightarrow & M \xrightarrow{f_1} & M_1 \longrightarrow \\ & \downarrow f_2 & \\ & M_2 & \\ & \downarrow & \end{array}$$

eine Fallunterscheidung, dh eine TM, die vom Endzustand f_1 von M nach M_1 übergeht, und von f_2 aus nach M_2.

Die folgende TM nennen wir „Band=0?“ (bzw „Band $i = 0$?“):

$$\begin{aligned} \delta(q_0, 0) &= (q_0, 0, R) \\ \delta(q_0, \Box) &= (ja, \Box, L) \end{aligned}$$

Sind f_1 und f_2 Endzustände von M so bezeichnet

$$\begin{array}{lcl} \longrightarrow & M \xrightarrow{f_1} & M_1 \longrightarrow \\ & \downarrow f_2 & \\ & M_2 & \\ & \downarrow & \end{array}$$

eine Fallunterscheidung, dh eine TM, die vom Endzustand f_1 von M nach M_1 übergeht, und von f_2 aus nach M_2.

Die folgende TM nennen wir „Band=0?“ (bzw „Band $i = 0$?“):

$$\begin{aligned} \delta(q_0, 0) &= (q_0, 0, R) \\ \delta(q_0, \Box) &= (ja, \Box, L) \\ \delta(q_0, a) &= (nein, a, N) \qquad \text{für } a \neq 0, \Box \end{aligned}$$

Sind f_1 und f_2 Endzustände von M so bezeichnet

$$\longrightarrow \quad M \xrightarrow{f_1} M_1 \longrightarrow$$
$$\downarrow f_2$$
$$M_2$$
$$\downarrow$$

eine Fallunterscheidung, dh eine TM, die vom Endzustand f_1 von M nach M_1 übergeht, und von f_2 aus nach M_2.

Die folgende TM nennen wir „Band=0?“ (bzw „Band $i = 0$?“):

$$\begin{aligned} \delta(q_0, 0) &= (q_0, 0, R) \\ \delta(q_0, \Box) &= (ja, \Box, L) \\ \delta(q_0, a) &= (nein, a, N) \qquad \text{für } a \neq 0, \Box \end{aligned}$$

wobei ja und $nein$ Endzustände sind.

Analog zur Fallunterscheidung kann man auch eine TM für eine Schleife konstruieren

$$\longrightarrow \quad \text{Band } i = 0? \xrightarrow{ja}$$
$$\uparrow\downarrow nein$$
$$M$$

Analog zur Fallunterscheidung kann man auch eine TM für eine Schleife konstruieren

$$\longrightarrow \quad \text{Band } i = 0? \xrightarrow{ja}$$
$$\uparrow\downarrow nein$$
$$M$$

die sich wie `while` Band $i \neq 0$ `do` M verhält.

Analog zur Fallunterscheidung kann man auch eine TM für eine Schleife konstruieren

$$\longrightarrow \quad \text{Band } i = 0? \xrightarrow{ja}$$
$$\uparrow\downarrow nein$$
$$M$$

die sich wie `while` Band $i \neq 0$ `do` M verhält.

Moral: Mit TM kann man imperativ programmieren:

`:=`
`;`
`if`
`while`

Analog zur Fallunterscheidung kann man auch eine TM für eine Schleife konstruieren

$$\longrightarrow \quad \text{Band } i = 0? \xrightarrow{ja}$$
$$\uparrow\downarrow nein$$
$$M$$

die sich wie `while` Band $i \neq 0$ `do` M verhält.

Moral: Mit TM kann man imperativ programmieren:

```
X := X±1 ; X:=y ; X:=w ;
;
if (X≠0)
while (X≠0)
```

4.4 LOOP-, WHILE- und GOTO-Berechenbarkeit

LOOP, WHILE $\equiv$ strukturierte Programme mit `for`/`while`-Schleifen

4.4 LOOP-, WHILE- und GOTO-Berechenbarkeit

LOOP, WHILE $\equiv$ strukturierte Programme mit `for`/`while`-Schleifen

GOTO $\equiv$ Assembler

4.4 LOOP-, WHILE- und GOTO-Berechenbarkeit

LOOP, WHILE $\equiv$ strukturierte Programme mit `for`/`while`-Schleifen

GOTO $\equiv$ Assembler

Syntax von LOOP-Programmen:

$$\begin{aligned} P \;\rightarrow\; & X \texttt{ := } X \texttt{ + } C \\ \mid\; & X \texttt{ := } X \texttt{ - } C \\ \mid\; & P\texttt{; } P \\ \mid\; & \texttt{LOOP } X \texttt{ DO } P \texttt{ END} \end{aligned}$$