

Grundlagen der Informatik

- Einführung in Berechenbarkeit und Komplexität -

Prof. Dr. Klaus Volbert



Hochschule für angewandte Wissenschaften
Fakultät Informatik und Mathematik

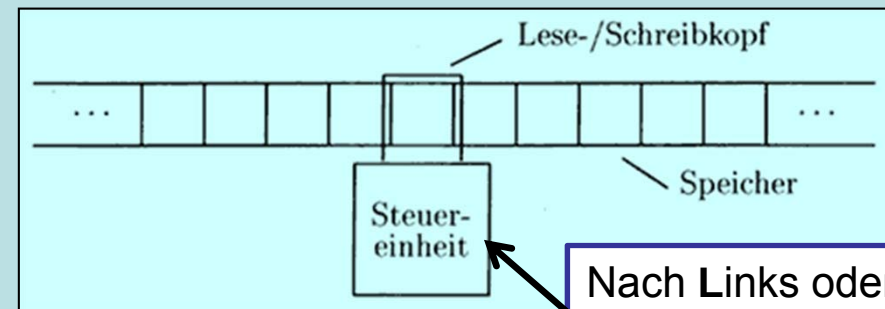
Wintersemester 2010/11
Regensburg, 15./16. Dezember 2010

Turingmaschine (Alan M. Turing, 1936)

- Eine deterministische Turingmaschine ist beschrieben durch ein 7-Tupel $TM = (Q, \Sigma, \Gamma, \delta, q_0, \#, F)$ mit:

- Q ist eine endliche, nichtleere Menge von **Zuständen**
- $\Sigma \subseteq \Gamma$ ist ein endliches, nicht leeres **Eingabealphabet**
- Γ ist ein endliches, nicht leeres **Bandalphabet**
- $\delta: Q \setminus F \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$ ist die **Übergangsfunktion**
- $q_0 \in Q$ ist der **Startzustand**
- $\# \in \Gamma \setminus \Sigma$ ist das **Blank-Symbol**

(leeres Feld, Initialwert)



Nach Links oder
Rechts bewegen
oder Nichts tun!

- $F \subseteq Q$ ist die Menge der **akzeptierenden Endzustände**

- **Akzeptanz**

- Eine TM **akzeptiert** eine Eingabe $x_1, \dots, x_n$, wenn gilt:

$$q_0 x_1, \dots, x_n \xrightarrow{*} \alpha q \beta \text{ mit } q \in F \text{ und } \alpha, \beta \in \Gamma^*$$

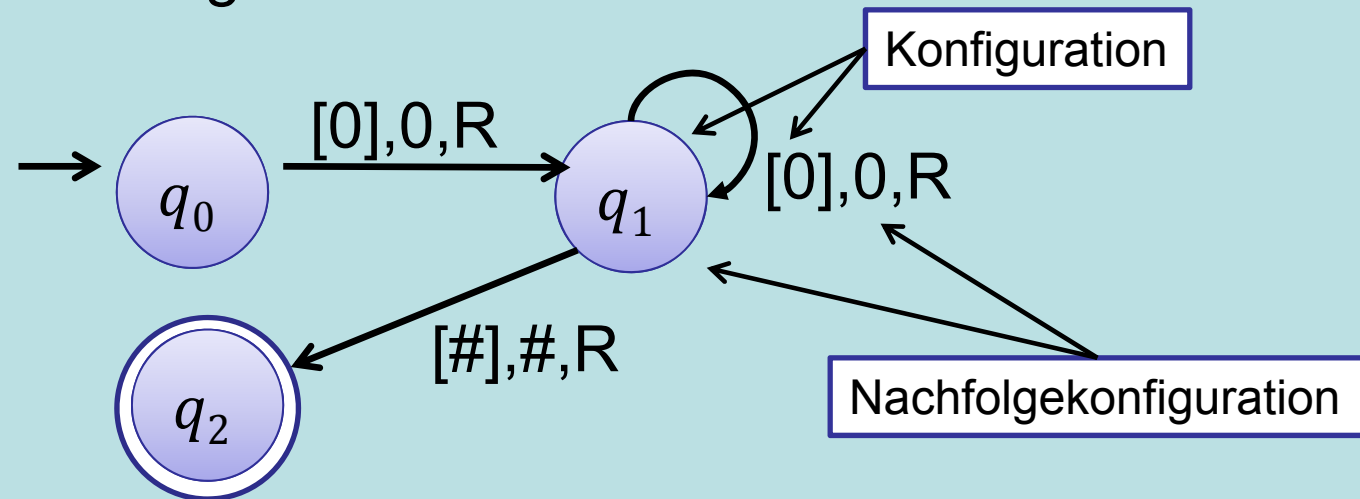
Beispiel (Turingmaschine verstehen)

- $TM = (Q, \Sigma, \Gamma, \delta, q_0, \#, F)$ mit

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0,1\}, \Gamma = \{\#, 0,1\}$
- $F = \{q_2\}$

δ	q_0	q_1	q_2
#	-	$(q_2, \#, R)$	-
0	$(q_1, 0, R)$	$(q_1, 0, R)$	-
1	-	-	-

- Grafische Darstellung:



- Die TM hält, wenn keine Kante vorhanden ist ("- in Tabelle)
(Beachte: DFA musste vollständig sein!)
- Welche Sprache akzeptiert die TM? $L = \{0^n \mid n \geq 1\}$

Beispiel (Turingmaschine entwerfen)

$$L = \{ 1^n 0^n \mid n \geq 1 \}$$

- Idee:
 - Prüfe, ob die Eingabe aus $1\dots 10\dots 0$ besteht
 - Wenn ja, dann prüfe ob (Anzahl Einsen) = (Anzahl Nullen)

δ	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
#	-	$(q_2, \#, L)$	-	$(q_4, \#, R)$	-	$(q_7, \#, N)$	$(q_2, \#, L)$	-
0	$(q_1, 0, R)$	$(q_1, 0, R)$	$(q_3, \#, L)$	$(q_3, 0, L)$	-	$(q_6, 0, R)$	$(q_6, 0, R)$	-
1	$(q_0, 1, R)$	-	-	$(q_3, 1, L)$	$(q_5, \#, R)$	$(q_6, 1, R)$	$(q_6, 1, R)$	-

↑
 Zunächst nur 1en

↑
 Dann nur 0en

↑
 Rechteste 0 streichen

↑
 Ganz nach links

↑
 Linkeste 1 streichen

↑
 Ganz nach rechts und Schleife,
wenn noch Zahlen da sind,
sonst akzeptieren

- TM = $(Q, \Sigma, \Gamma, \delta, q_0, \#, F)$ mit $Q = \{q_0, q_1, \dots, q_7\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{\#, 0, 1\}$, $F = \{q_7\}$

Turing-Berechenbarkeit

- Eine (partielle) Funktion $f: \mathbb{N}^k \rightarrow \mathbb{N}$ heißt **Turing-berechenbar**, falls es eine (deterministische) Turingmaschine TM gibt, die bei Eingabe von $x_1, \dots, x_n$ die Ausgabe $f(x_1, \dots, x_n)$ liefert und hält, d.h.

$$q_0 x_1, \dots, x_n \xrightarrow{*} qy \text{ mit } q \in F \text{ und } y \in \Gamma^* \text{ und } y = f(x_1, \dots, x_n)$$

- Falls TM nicht hält, ist $f(x_1, \dots, x_n)$ nicht definiert
- TM-äquivalente Berechenbarkeitsmodelle
 - Registermaschinen (RAM)
 - Goto-Programme (IF...GOTO...)
 - While-Programme (IF...THEN..., WHILE ...DO ...)
 - μ -rekursive Funktionen

} Beweis durch
Simulation
- Ausdrucksschwächere Berechenbarkeitsmodelle
 - Loop-Programme (IF...THEN..., FOR ...TO ... DO...)
 - Primitiv-rekursive Funktionen

} äquivalent

Church'sche These (1936, auch Church-Turing-These)

- These: Die Klasse der Turing-berechenbaren Funktionen ist genau die Klasse der intuitiv berechenbaren Funktionen
- Alternative Formulierungen/Folgerungen
 - Jede Funktion, die überhaupt in irgendeiner Weise berechenbar ist, kann durch eine Turingmaschine berechnet werden
 - Jedes Problem, das überhaupt maschinell lösbar ist, kann von einer Turingmaschine gelöst werden
- Anmerkungen
 - Church'sche These ist nicht beweisbar, da „intuitiv berechenbare Funktionen“ nicht formalisiert werden können
 - Anerkanntes Rechenmodell: Von-Neumann-Rechner
 - Idealisierte Von-Neumann-Rechner: Registermaschinen
 - Registermaschinen sind äquivalent zu Turingmaschinen
- Church'sche These gilt als allgemein akzeptiert

- Interpretation der Church-Turing-These:
 - Bisher und in Zukunft vorgenommene „vernünftige“ Definitionen von Algorithmus sind gleichwertig und haben die gleiche Bedeutung wie die bisher bekannten Definitionen!
- Algorithmus = Programm für eine TM
 - = Programm für eine RAM
 - = Programm für andere Modelle
(Goto, While, μ -Rekursion)
 - = Programm in C/C++
 Pascal
 Java
 C#, ...

Akzeptanz und Entscheidbarkeit

- Im Gegensatz zu einem DFA kann es bei einer TM zu einer Endlosschleife kommen
 - Erinnerung DFA: Endliche Eingabe wird einmal von links nach rechts gelesen und verarbeitet, d.h. die Verarbeitung terminiert
- Akzeptanz
 - Eine TM M **akzeptiert** eine Sprache L , falls M alle $x \in L$ akzeptiert (d.h. M gestartet mit $x \in L$ hält in einem akzeptierenden Endzustand)
 - Anmerkung: Es kann $x \notin L$ existieren mit M gestartet mit x hält nicht
- Entscheidbarkeit
 - Eine TM M **entscheidet** eine Sprache L , falls M die Sprache L akzeptiert und für alle $x \notin L$ nach endlich vielen Schritten in einem nicht akzeptierenden Endzustand hält
- Eine Sprache L heißt
 - rekursiv **aufzählbar** (semi-entscheidbar) $\Leftrightarrow$ Es gibt eine TM, die L akzeptiert
 - rekursiv oder **entscheidbar** $\Leftrightarrow$ Es gibt eine TM, die L entscheidet