

Grundlagen Theoretischer Informatik 2

WiSe 2009/10 in Trier

Henning Fernau

Universität Trier

fernau@uni-trier.de

Grundlagen Theoretischer Informatik 2 Gesamtübersicht

- Organisatorisches; Einführung
- Ersetzungsverfahren: [Grammatiken und Automaten](#)
- Rekursionstheorie-Einführung
- Komplexitätstheorie-Einführung

Organisatorisches in nächster Zeit

Nächste Vorlesungsstunde:

Es findet eine Vorrechen- und Fragestunde statt

DENN

In der letzten Vorlesung vor Weihnachten schreiben wir unsere (also: Sie Ihre...)

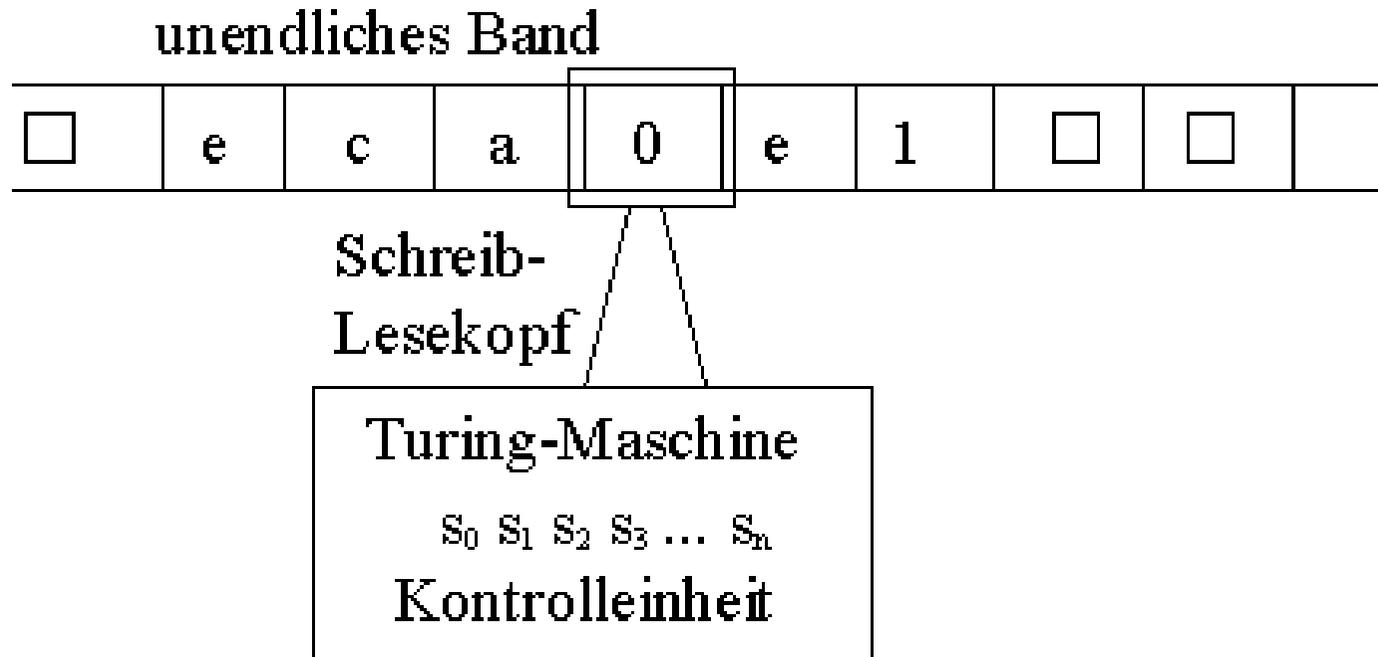
Zwischenklausur !

Grundregel: Sie bringen Stifte und Studentenausweis mit, lassen aber am besten alles andere daheim!

Alan Turing

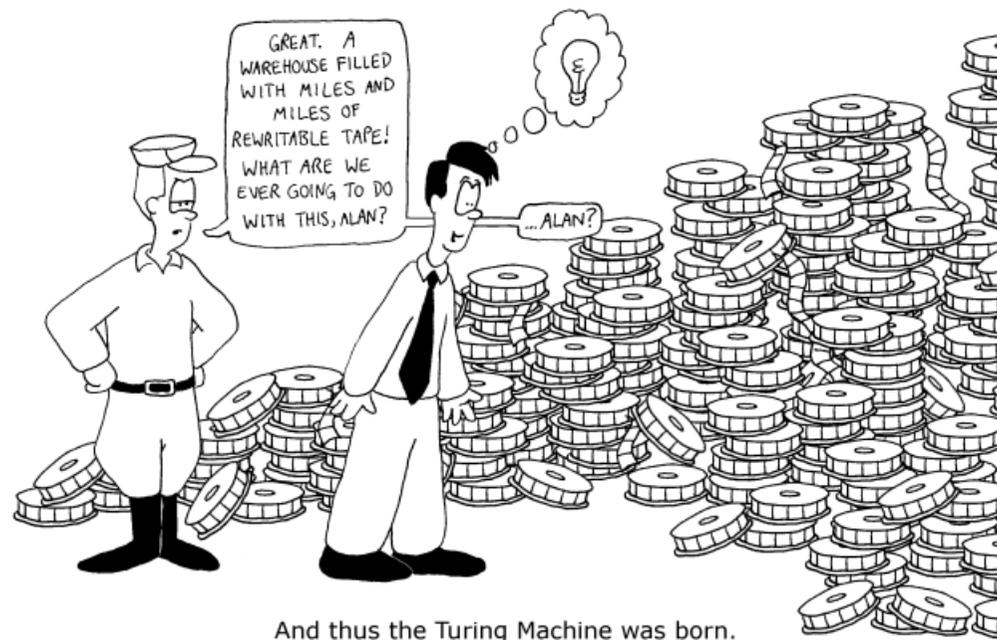


Turingmaschinen



Grundidee Turings (1936): Simulation eines **menschlichen Computers**.

Turingmaschinen



Turingmaschinen: formaler

Eine *Turingmaschine* (TM) ist durch ein 7-Tupel beschrieben:

$$TM = (S, E, A, \delta, s_0, \square, F)$$

Dabei bedeuten

- $S = \{s_0, s_1, \dots, s_n\}$ die Menge der *Zustände*,
- $E = \{e_1, e_2, \dots, e_r\}$ das endliche *Eingabealphabet*,
- $A = \{a_0, a_1, \dots, a_m\}$ das endliche *Arbeitsalphabet* (auch Bandalphabet genannt), es sei dabei $E \subset A$,

- s_0 der *Startzustand*,
- $\alpha_0 = \square$ das *Blank-Symbol*, das zwar dem Arbeitsalphabet, aber nicht dem Eingabealphabet angehört,
- $F \subseteq S$ die Menge der *Endzustände*,
- δ sei die *Überföhrungsfunktion/-relation* mit (im deterministischen Fall)

$$\delta : (S \setminus F) \times A \rightarrow S \times A \times \{L, R, N\}$$

bzw. (im nichtdeterministischen Fall) $\delta \subseteq ((S \setminus F) \times A) \times (S \times A \times \{L, R, N\})$.
 Hier bedeutet: L= links, R = rechts, N= neutral.

Turingmaschinen: Dynamik

$\delta(s, a) = (s', b, x)$ bzw. $((s, a), (s', b, x)) \in \delta$ (mit $x \in \{L, R, N\}$) habe folgende Bedeutung:

Wenn sich der Automat im Zustand s befindet und unter dem Kopf das Zeichen a steht, so schreibt der Automat b und geht ein Feld nach rechts (R), bzw. links (L), bzw. bewegt sich nicht (N) und geht in den Zustand s' über.

Wie schon bei anderen Automatenmodellen gesehen, kann δ in Form einer Überführungstafel notiert werden. In jedem Kästchen stehen dann ein oder mehrere Tripel, bestehend aus neuem Zustand, geschriebenem Zeichen und Positionszeichen.

Eine **Konfiguration** einer Turingmaschine $TM = (S, E, A, \delta, s_0, \square, F)$ ist ein Tripel (u, s, v) aus $A^* \times S \times A^+$:

- uv ist aktuelle Bandinschrift
- s ist der aktuelle Zustand.
- Schreib-Lesekopf ber erstem Zeichen von v , daher $v \neq \lambda$.
- Start der Maschine: $v \in E^* \cup \{\square\}$ (Eingabe), $s = s_0$, $u = \lambda$.
- O.B.d.A. $S \cap A = \emptyset$, daher Schreibweise usv statt (u, s, v)

Übergangsrelation \vdash

Zu einer (nicht)deterministischen Turingmaschine

$$TM = (S, E, A, \delta, s_0, \square, F)$$

wird die Übergangsrelation \vdash aus $(A^* \times S \times A^+)^2$ folgendermaßen definiert:

- $a_1 \dots a_m s b_1 \dots b_n \vdash a_1 \dots a_m s' c b_2 \dots b_n$ mit $((s, b_1), (s', c, N)) \in \delta, m \geq 0, n \geq 1$
- $a_1 \dots a_m s b_1 \dots b_n \vdash a_1 \dots a_m c s' b_2 \dots b_n$ mit $((s, b_1), (s', c, R)) \in \delta, m \geq 0, n \geq 2$
- $a_1 \dots a_m s b_1 \dots b_n \vdash a_1 \dots a_{m-1} s' a_m c b_2 \dots b_n$ mit $((s, b_1), (s', c, L)) \in \delta, m \geq 1, n \geq 1$

Bem.: Die Konfigurationen fgen nach Bedarf Blanks an die Teilwörter v bzw. u an.

Inkrementieren einer Binärzahl

Beispiel:

$$TM = (\{s_0, s_1, s_2, s_f\}, \{0, 1\}, \{0, 1, \square\}, \delta, s_0, \square, \{s_f\})$$

mit

$\delta(s, a)$	0	1	\square
s_0	$(s_0, 0, R)$	$(s_0, 1, R)$	(s_1, \square, L)
s_1	$(s_2, 1, L)$	$(s_1, 0, L)$	$(s_f, 1, N)$
s_2	$(s_2, 0, L)$	$(s_2, 1, L)$	(s_f, \square, R)
s_f	—	—	—

Beispiel:

$$\begin{aligned} s_0 111 &\vdash 1s_0 11 \vdash 11s_0 1 \vdash 111s_0 \square \\ &\vdash 11s_1 1 \square \vdash 1s_1 10 \square \vdash s_1 100 \square \vdash s_1 \square 000 \square \\ &\vdash s_f 1000 \square \end{aligned}$$

Arbeitsweise:

- (1) Die Maschine läuft solange nach rechts, bis sie das Ende der Zahl (Ziffer mit kleinstem Gewicht) erreicht hat.
- (2) Ein weiterer Schritt führt auf ein Blank (wenn zu Beginn nichts auf dem Band steht, steht der Lese-Schreibkopf schon auf einem Blank).
- (3) Dann erfolgt wieder ein Schritt nach links und Übergang nach s_1 . \rightsquigarrow Arbeitsposition erreicht.
- (4A) Trifft sie auf eine Null, so invertiert sie sie (Addition von 1) und geht nach s_2 .
- (4B) Trifft sie vorher auf Einsen, müssen diese invertiert werden.
- (4C) Spezialfall: keine 0 in der Zahldarstellung; dann wächst die Stellenzahl um 1 und die Maschine schreibt eine 1 anstelle des ersten linken Blanks.
- (5) Ist die führende Stelle erreicht, geht die Maschine in den Endzustand über und der Kopf bleibt hier stehen.

Initialkonfiguration, Finalkonfiguration, akzeptierte Sprache

- *Initialkonfiguration* beim Start der Turingmaschine mit Eingabe $w \in E^*$ ist s_0w (bzw. $s_0\Box$, falls $w = \lambda$).
- *Finalkonfigurationen* sind alle Konfigurationen us_fv mit $s_f \in F$. Hier kann die Berechnung nicht mehr fortgesetzt werden.
- Weiter ist

$$L(TM) := \{w \in E^* \mid s_0w \vdash^* us_fv, s_f \in F, u, v \in A^*\}$$

die *von der Turingmaschine akzeptierte Sprache* L .

Hinweis: TM-Simulatoren

Einige Beobachtungen

Simulation endlicher Automat durch Turingmaschine z.B. wie folgt:

- Schreib-Lesekopf hat ausschließlich lesende Funktion
- Lesekopf zeichenweise lesend nach rechts
- Zustandsübergänge des endlichen Automaten übernommen
- Akzeptieren bei Erreichen von \square (d.h. Ende der Eingabe) im 'Automaten-Endzustand'

Simulation Kellerautomat durch Turingmaschine z.B. wie folgt:

- Turing-Band zweigeteilt: Rechts Eingabewort, links Keller
- Übergangsfunktion des NKA durch mehrere Schritte der TM
- Gelesene Zeichen der Eingabe mit Spezialzeichen markieren
- Bestimmung des Überganges im NKA durch Inspektion der noch nicht markierten Eingabe und des simulierten Kellers

⇒ Turingmaschinen mindestens so mächtig wie Kellerautomaten

Turingmaschinen als Ersetzungssysteme

Die formalen Einzelheiten überlasse ich diesmal Ihnen . . .

Beachte: Die “Konfigurationsübergänge” entsprechen wieder Schritten eines Ersetzungssystems.

Turingmaschine zu $L = \{a^n b^n c^n \mid n > 0\}$

TM = (S, E, A, δ , s_0 , \square , F) mit

$$S = \{s_0, s_1, s_2, s_3, s_4, s_f\}$$

$$A = \{a, b, c, 0, 1, 2, \square\}$$

$$F = \{s_f\}$$

Idee der Arbeitsweise:

- Ersetze jeweils erstes a durch 0, b durch 1 und c durch 2
- Achte dabei auf korrekte Reihenfolge
- Teste am Ende, ob alle a, b, c ersetzt wurden

$$s_0 a^n b^n c^n \vdash^* 0^k s_0 a^m 1^k b^m 2^k c^m \vdash^* 0^n s_0 1^n 2^n \vdash^* 0^n 1^n 2^n s_f \square$$

Die zugehörige Übergangstabelle

	a	b	c	0	1	2	□
s_0	$(s_1, 0, R)$				$(s_4, 1, R)$		
s_1	(s_1, a, R)	$(s_2, 1, R)$			$(s_1, 1, R)$		
s_2		(s_2, b, R)	$(s_3, 2, L)$			$(s_2, 2, R)$	
s_3	(s_3, a, L)	(s_3, b, L)		$(s_0, 0, R)$	$(s_3, 1, L)$	$(s_3, 2, L)$	
s_4					$(s_4, 1, R)$	$(s_4, 2, R)$	(s_f, \square, N)
s_f							

Ist die Übergangsfunktion nicht definiert, bleibt der Automat stehen.

Also: Turingmaschinen sind stärker als Kellerautomaten!

Beispiel: $s_0abcc \vdash 0s_1bcc \vdash 01s_2cc \vdash 0s_312c \vdash s_3012c \vdash 0s_012c \vdash 01s_42c \vdash 012s_4c$ halt

These von Church / Turing

Turingmaschinen können “alles”, was behauptet jemals von “Computern” gemacht werden kann.

Diese These sieht man durch Überlegung ein (Bsp.: Typ-0 Sprachen können von TM akzeptiert werden), sie lässt sich aber **nicht beweisen**.

Möchten Sie aber Ihre Programme in Zukunft in TM-Notation schreiben ?!

Typ-0 Sprachen lassen sich durch TM-Akzeptanz kennzeichnen I

Eine Phrasenstrukturgrammatik G_T , die eine TM M simulieren soll, arbeitet wie folgt:

- (1) Es wird die Sprache " $s_0w\$w$ " generiert für beliebige Eingabewörter w der Turingmaschine.
- (2) Die Arbeitsweise der Turingmaschine (Konfigurationsübergänge) wird nun auf dem ersten Wortteil simuliert.
- (3) Wird eine Finalkonfiguration erreicht, so besteht die Möglichkeit, den ersten Wortteil und den Trenner $\$$ zu löschen und so w zu generieren.

Typ-0 Sprachen lassen sich durch TM-Akzeptanz kennzeichnen II

Eine TM M_G , die eine Phrasenstrukturgrammatik G simuliert, arbeitet “akzeptierend.”

- (1) Ein Wort w stehe eingangs auf dem Band ($w \in L(G)$ ist unbekannt)
- (2) Forlaufend sucht M_G eine beliebige Regel $\alpha \rightarrow \beta$ der Grammatik “rückwärts” anzuwenden:
Es wird also eine Stelle im augenblicklichen Wort auf dem Band gesucht, an der β steht, und dann wird β durch α ersetzt.
Dabei sind evtl. Bandteile auseinander- oder zusammenschieben.
- (3) Wird die Satzform, die nur aus dem Startsymbol besteht, erreicht, hält und akzeptiert die TM.

Linear beschränkte Automaten

Eine TM heißt *linear beschränkter Automat* (LBA), wenn sie keine Blankzeichen überschreiben darf.

Satz: L ist Typ-1 gdw. L wird von LBA akzeptiert.

Beweis: ... ähnlich wie auf der letzten Folie ...

Eine Normalform für Grammatiken (Typ-0, Typ-1, Typ-2)

Satz: Zu jeder Grammatik gibt es eine äquivalente gleichen Typs, bei der alle Regeln mit Terminalzeichen a von der Form $A \rightarrow a$ sind.

Beweis: Es sei $G = (N, \Sigma, P, S)$ eine Grammatik.

Für jedes Terminalzeichen a führe ein neues Nichtterminalzeichen X_a ein.

$M := \{X_a \mid a \in \Sigma\}$; $N' = N \cup M$.

Definiere Morphismus $h : (\Sigma \cup N)^* \rightarrow (M \cup N)^*$: $h(a) = X_a$ für $a \in \Sigma$ und $h(A) = A$ für $A \in N$.

$P' := \{h(\alpha) \rightarrow h(\beta) \mid \alpha \rightarrow \beta \in P\} \cup \{X_a \rightarrow a \mid a \in \Sigma\}$.

In $G' := (N', \Sigma, P', S)$ kann eine Ableitung von G simuliert werden:

1. Phase: Simulation auf "Nichtterminalebene"
2. Phase: Verwandlung der X_a in entsprechende a -Terminale.

Da keine kontextabhängigen Regeln mit Terminalen auf der linken Seite existieren, kann jede terminierende Ableitung, in der $X_a \rightarrow a$ -Regeln vor eigentlichen Simulationsregeln angewendet werden, umgeordnet werden in 1. und 2. Phase.

Abschlusseigenschaften bei Typ-0/-1 Sprachen (Auswahl)

Vereinigung / Konkatenation: “Standardkonstruktion” wie Typ-2

Durchschnitt: NF-Grammatiken G_1 und G_2 werden simuliert auf “Produktalphabet” $N_1 \times N_2$. Nur für Paare (X_a, X_a) gibt es terminierende Regeln $(X_a, X_a) \rightarrow a$.

Komplement: Typ-1 Sprachen sind abgeschlossen; die entsprechende Technik des induktiven Zählens ist Gegenstand der Komplexitätstheorie.
Typ-0 Sprachen sind nicht abgeschlossen (Satz von Post . . . nach Weihnachten).

Was leistet Nichtdeterminismus?

- bei endlichen Automaten:
DEAs akzeptieren dieselben Sprachen wie NEAs
(Potenzautomatenkonstruktion)
- bei Kellerautomaten:
DKAs können weniger als NKAs (Mitteilung)
- bei LBAs:
Es ist ein offenes Problem, ob DLBAs dasselbe können wie LBAs.
- bei TMs:
DTMs können dasselbe wie NTMs:
Simuliere NTM durch DTM durch "Breitensuche" im Konfigurationsgraph