Grundlagen Theoretischer Informatik 2

WiSe 2011/12 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

$\boxed{\text{Grundlagen Theoretischer Informatik 2}}$ Gesamtübersicht

- Organisatorisches; Einführung
- Ersetzungsverfahren: Grammatiken und Automaten
- Rekursionstheorie-Einführung
- Komplexitätstheorie-Einführung

Organisatorisches in nächster Zeit

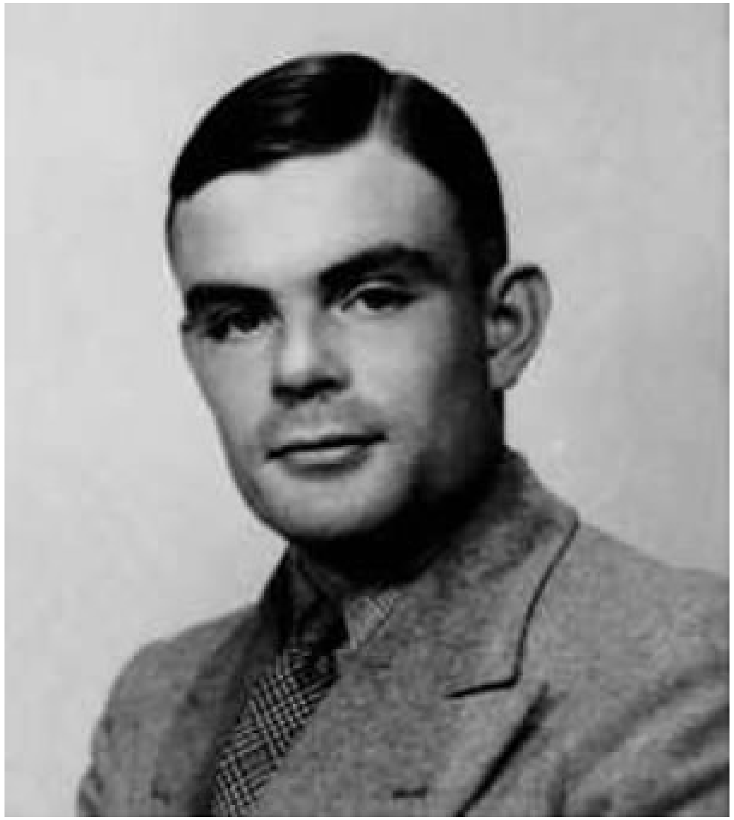
Bitte nehmen Sie unsere Sprechstunden wahr

DENN

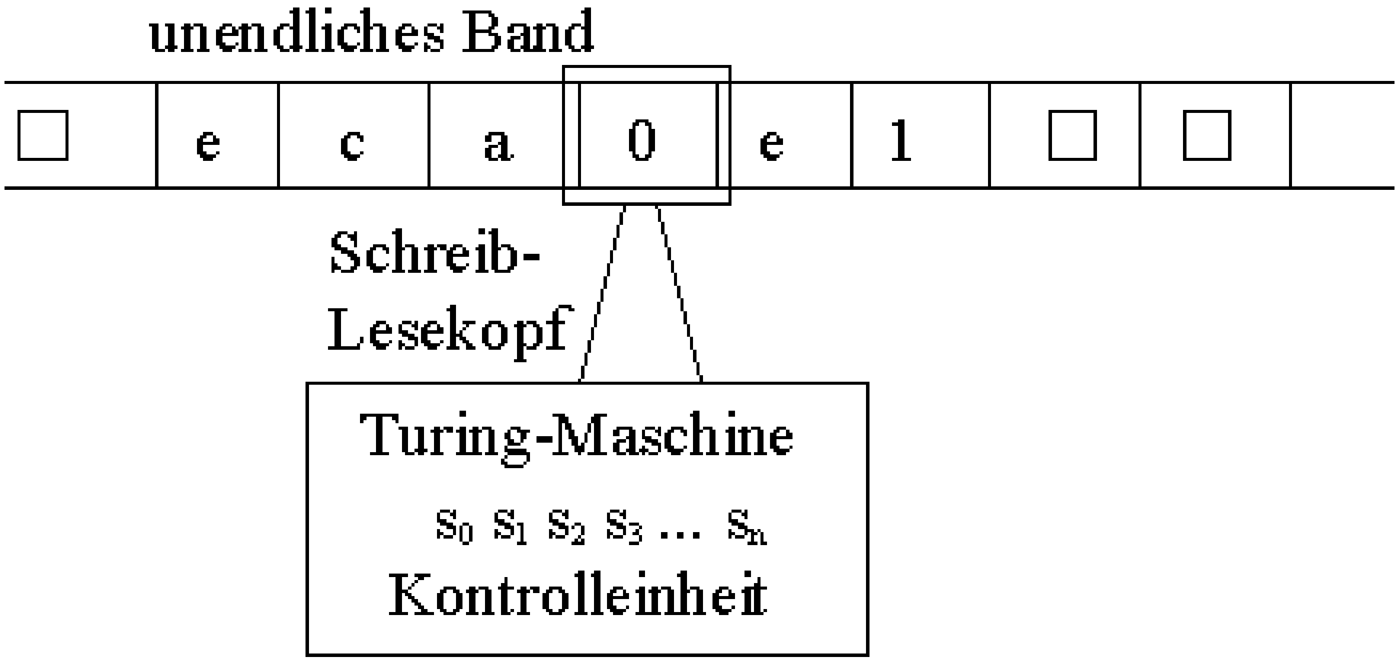
In der letzten Vorlesungsstunde vor Weihnachten schreiben wir unsere (also: Sie Ihre. . .) Zwischenklausur !

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

And thus the Turing Machine was born.

Turingmaschinen: formaler

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

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

Dabei bedeuten

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

- s_0 der *Startzustand*,

- $a_0 = \Box$ das *Blank-Symbol*, das zwar dem Arbeitsalphabet, aber nicht dem Eingabealphabet angehört,

- $F \subseteq S$ die Menge der *Endzustände*,

- δ sei die *Überführungsfunktion/-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

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

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

- $u_1...u_p s v_1...v_q \vdash u_1...u_p s' c v_2...v_q$ mit $((s, v_1), (s', c, N)) \in \delta, p \geq 0, q \geq 1$
- $u_1...u_p s v_1...v_q \vdash u_1...u_p c s' v_2...v_q$ mit $((s, v_1), (s', c, R)) \in \delta, p \geq 0, q \geq 2$
- $u_1...u_p s v_1...v_q \vdash u_1...u_{p-1} s' u_p c v_2...v_q$ mit $((s, v_1), (s', c, L)) \in \delta, p \geq 1, q \geq 1$

Bem.: Die Konfigurationen fügen nach Bedarf links bzw. rechts Blanksymbole an die Teilwörter $u = u_1 \dots u_p$ bzw. $v = v_1 \dots v_q$ 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:

$$\begin{aligned} s_0111 \quad & \vdash\ 1s_011 \ \vdash\ 11s_01 \ \vdash\ 111s_0\square \\ & \vdash\ 11s_11\square \ \vdash\ 1s_110\square \ \vdash\ s_1100\square \ \vdash\ s_1\square000\square \\ & \vdash\ s_f1000\square \end{aligned}$$

Arbeitsweise:
(1) Die Maschine läuft solange nach rechts, bis sie das Ende der Zahl (Einerstelle) 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 geht 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 $s_0 w$ (bzw. $s_0 \square$, falls $w = \lambda$).

- *Finalkonfigurationen* sind alle Konfigurationen $u s_f v$ mit $s_f \in F$. Hier kann die Berechnung nicht mehr fortgesetzt werden.

- Weiter ist

$$L(TM) := \{w \in E^* \mid s_0 w \vdash^* u s_f v, s_f \in F, u, v \in A^*\}$$

die *von der Turingmaschine akzeptierte Sprache* L.

Hinweis: TM-Simulatoren im Internet

Einige Beobachtungen I

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'

Einige Beobachtungen II

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

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

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

$$\begin{aligned} S &= \{s_0, s_1, s_2, s_3, s_4, s_f\} \\ A &= \{a, b, c, 0, 1, 2, \Box\} \\ F &= \{s_f\} \end{aligned}$$

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

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

Die zugehörige Übergangstabelle

	a	b	c	0	1	2	$\Box$
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					$(s_4, 1, R)$	$(s_4, 2, R)$	$(s_f, \Box, N)$
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 berhaupt 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 Typ-0-Grammatik G_T, die eine TM M simulieren soll, arbeitet wie folgt:

(1) Es wird die Sprache "$s_0 w \$ w$" 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 Typ-0-Grammatik G simuliert, arbeitet “akzeptierend.”

(1) Ein Wort w stehe eingangs auf dem Band ($w \in L(G)$ ist unbekannt)
(2) Fortlaufend sucht M_G eine beliebige Regel $\alpha \to \beta$ 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 zusammenzuschieben.
(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: $\mathbb{L}$ ist Typ-1 gdw. $\mathbb{L}$ wird von LBA akzeptiert.

Beweis: …ähnlich wie auf der letzten Folie …

Morphismen

Sind $(M, \circ, e)$ und $(N, \Box, 1)$ Monoide, so heißt $h : M \to N$ *Morphismus* gdw:
1. $h(e) = 1$ und 2. $\forall x, y \in M : h(x \circ y) = h(x) \Box h(y)$.
Morphismen sind strukturerhaltende Abbildungen.
Aus der Linearen Algebra dürften Sie Vektorraum-(Homo-)Morphismen kennen.

Für uns interessant: frei erzeugte Monoide $(X^*, \cdot, \lambda)$ und $(Y^*, \cdot, \lambda)$.
Da sich jedes Wort eindeutig als Buchstabenfolge darstellen lässt, gilt:
Satz: Ein Morphismus $h : X^* \to Y^*$ ist eindeutig festgelegt durch Angabe von $h(a)$ für alle $a \in X$.

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 From $A \to a$ sind.

Beweis: Es sei $G = (N, \Sigma, P, S)$ eine Grammatik vom Typ i, $i = 0, 1, 2$.

Für jedes Terminalzeichen a führe ein neues Nichtterminalzeichen X_a ein.

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

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

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

$G' := (N', \Sigma, P', S)$ ist ebenfalls vom Typ i.

In G' kann eine Ableitung von G simuliert werden:

1. Phase: Simulation auf “Nichtterminalebene”

2. Phase: Verwandlung der X_a in entsprechende a-Terminale.

Da in G' keine kontextabhängigen Regeln mit Terminalen auf der linken Seite existieren, kann jede terminierende Ableitung, in der $X_a \to a$-Regeln vor den eigentlichen Simulationsregeln angewendet werden, umgeordnet werden in die angegebenen Phasen.

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 $(X_a, X_a) \to a$.

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? (ein zentrales Thema der Informatik)

- 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