TECHNISCHE UNIVERSITÄT DRESDEN

FORMALE SYSTEME

18. Vorlesung: Turingmaschinen

Markus Krötzsch

Professur für Wissensbasierte Systeme

TU Dresden, 14. Dezember 2023

Zusammenfassung Typ 2

Kontextfreie Sprachen

- Beschrieben durch kontextfreie Grammatiken und Kellerautomaten
- Können mehrdeutig sein
- Können deterministisch sein (beschrieben durch deterministische Kellerautomaten und **LR**(k)-Grammatiken)
- Wortproblem polynomiell, im deterministischen Fall sogar linear
- Sonstige Entscheidungsprobleme meist viel schwerer als bei regulären Sprachen
- Nicht unter allen Operationen abgeschlossen

Turingmaschinen

Kellerautomaten erweitern

Unmittelbar denkbare Erweiterungen von Kellerautomaten:

- Verwendung von zwei oder mehr Kellern
- Verwendung von FIFO-Speichern (Warteschlange)
- Verwendung einer endlichen Menge von Zählern
- ...

All diese Modelle erkennen genau die Typ-0-Sprachen!

Das klassische Automatenmodell
für diese Sprachklasse ist die Turingmaschine.

Was kann diese Art von Automaten?
Alles, was überhaupt auf Computern machbar ist!


Alan Turing (publ. dom.)

Church-Turing-These: Die Turingmaschine kann alle Funktionen berechnen, die intuitiv berechenbar sind.

Turingmaschinen – Grundideen

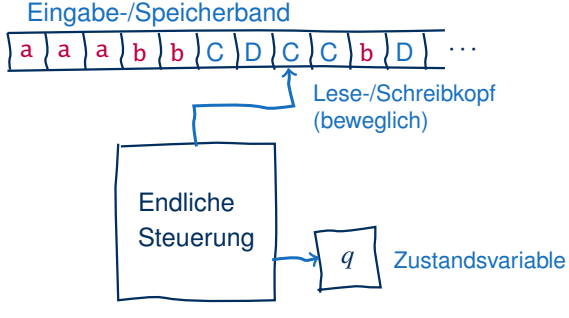
Wesentliche Designentscheidungen
bei der Gestaltung von Turingmaschinen (TMs):

- TMs haben eine endliche Steuerung (wie bei NFA und PDA)
- Es gibt eine unbeschränkte Menge an Speicher (wie bei PDA)
- Die TM kann in jedem Schritt ein Zeichen aus dem Speicher lesen und eines schreiben (wie bei PDA)
- Der Lese-/Schreibzugriff ist an jeder beliebigen Speicheradresse möglich (im Gegensatz zu PDA)

  Zur praktischen Implementierung speichert die TM die aktuelle Adresse und kann diese in jedem Schritt um eins erhöhen oder verringern
- Zur Vereinfachung wird die Eingabe einfach beim Start in den Speicher übergeben, so dass „Lesen der Eingabe“ und „Lesen aus Speicher“ die selbe Operation sind

Turingmaschinen – Grundideen (2)

Schematische Darstellung:



Übergangsfunktion:

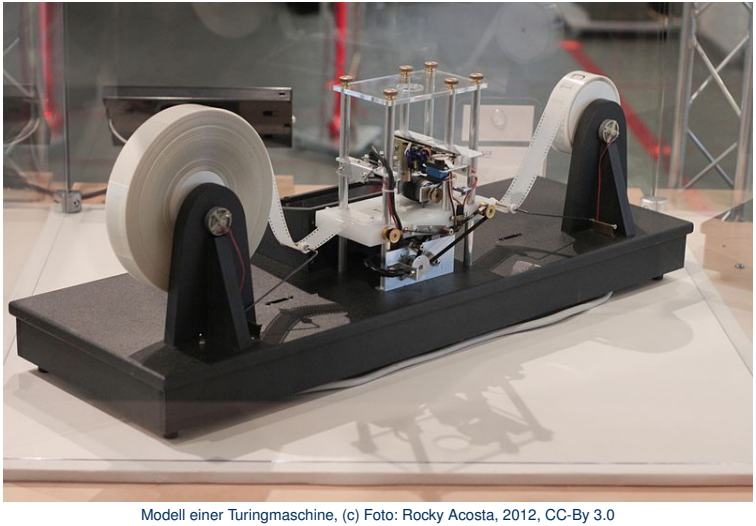
- Eingabe: aktueller Zustand, gelesenes Zeichen
- Ausgabe: neuer Zustand, geschriebenes Zeichen, Änderung Lese-/Schreibadresse ($\hat{=}$ Bewegung Lese-/Schreibkopf)

Speicherzugriff in TMs

Speicherverwaltung:

- Zu jedem Zeitpunkt verwendet die TM endlich viele Speicherzellen
- Speicherzellen als „Band“ von links nach rechts
- Lese-/Schreibkopf kann frei auf Band bewegt werden (pro Schritt um eine Zelle nach links oder rechts)
- Am linken Rand kann der Kopf nicht weiter nach links bewegt werden
- Am rechten Rand des Speichers kann der Kopf nach rechts bewegt werden: dann wird dort eine neue Speicherzelle mit dem Inhalt ␣ (Leerzeichen, Blank) angefügt

Alternative Vorstellung: Der Speicher ist ein einseitig unendlich langes Band, auf dem am Anfang nur eine (endliche) Eingabe steht, gefolgt von unendlich vielen leeren Zellen (␣).



Modell einer Turingmaschine, (c) Foto: Rocky Acosta, 2012, CC-By 3.0

Definition TM

Eine (deterministische) Turingmaschine (DTM) ist ein Tupel $\mathcal{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$ mit den folgenden Bestandteilen:

- Q: endliche Menge von Zuständen
- Σ: Eingabealphabet
- Γ: Arbeitsalphabet mit $\Gamma \supseteq \Sigma \cup \{\sqcup\}$
- δ: Übergangsfunktion, eine partielle Funktion

$$Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$$

- q_0: Startzustand $q_0 \in Q$
- F: Menge von akzeptierenden Endzuständen $F \subseteq Q$

Dabei bedeutet $\delta(q, a) = \langle p, b, D \rangle$:

„Liest die TM in Zustand q unter dem Lese-/Schreibkopf ein a, dann wechselt sie zu Zustand p, überschreibt das a mit b und verschiebt den Lese-/Schreibkopf gemäß $D \in \{L, R, N\}$ (nach links, nach rechts, gar nicht).“

Beispiel (1)

Wir konstruieren eine TM mit dem Eingabealphabet $\Sigma = \{0\}$,
welche Wörter der Sprache $\{0^{2^i} \mid i \geq 0\}$ akzeptiert
(Ketten von 0, deren Länge eine Zweierpotenz ist)

Arbeitsweise:

(1) Laufe von links nach rechts über die Eingabe und ersetze dabei jede zweite 0 durch X

(2) Falls insgesamt nur eine 0 gefunden wird, akzeptiere

(3) Falls eine andere ungerade Zahl an 0 gefunden wird, verwirf

(4) Laufe zurück zum Anfang des Bandes

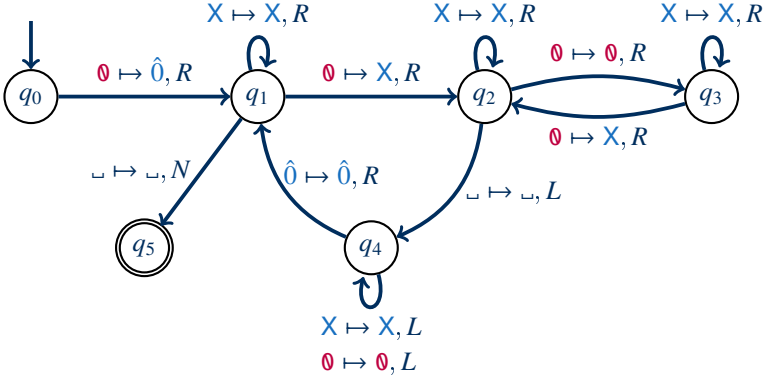
(5) Wiederhole die Schritte ausgehend von (1)

Beispiel (2)

Wir können TMs in Diagrammen darstellen:

Ein Pfeil $s_1 \mapsto s_2, D$ von q_1 nach q_2 bedeutet $\delta(q_1, s_1) = \langle q_2, s_2, D \rangle$

Übergangsrelation zum Beispiel:



Arbeitsweise einer TM: Übergänge

Sei $\mathcal{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$ eine DTM.

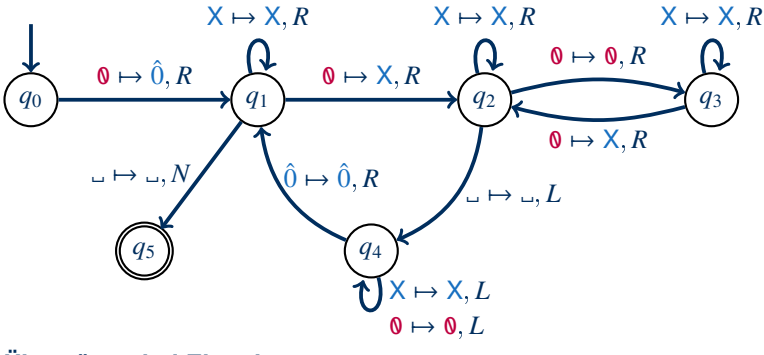
Eine Konfiguration von $\mathcal{M}$ ist ein Wort $w\,q\,v \in \Gamma^* \circ Q \circ \Gamma^*$, wobei w den Bandinhalt vor dem Lese-/Schreibkopf, q den aktuellen Zustand und v den Bandinhalt ab dem Lese-/Schreibkopf darstellt.

Sei $w\,q\,av$ eine Konfiguration, mit $w, v \in \Gamma^*$, $a \in \Gamma$ und $q \in Q$. Die Übergangsrelation $\vdash$ ist wie folgt definiert:

- Falls $\delta(q, a) = \langle r, b, N \rangle$, dann $w\,q\,av \vdash w\,r\,bv$.
- Falls $\delta(q, a) = \langle r, b, R \rangle$
  - falls $v \neq \epsilon$, dann $w\,q\,av \vdash wb\,r\,v$;
  - falls $v = \epsilon$, dann $w\,q\,av \vdash wb\,r\,\sqcup$.
- Falls $\delta(q, a) = \langle r, b, L \rangle$
  - falls $w = w'c$ mit $c \in \Gamma$, dann $w\,q\,av \vdash w'\,r\,cbv$;
  - falls $w = \epsilon$, dann $w\,q\,av \vdash r\,bv$.

Mit $\vdash^*$ bezeichnen wir den reflexiven, transitiven Abschluss von $\vdash$.

Beispiel (3)



Übergänge bei Eingabe von 0000:

$q_0\,0000 \vdash \hat{0}\,q_1\,000 \vdash \hat{0}X\,q_2\,00 \vdash \hat{0}X0\,q_3\,0 \vdash \hat{0}X0X\,q_2\,␣ \vdash \hat{0}X0\,q_4\,X␣ \vdash$

$\hat{0}X\,q_4\,0X␣ \vdash \hat{0}\,q_4\,X0X␣ \vdash q_4\,\hat{0}X0X␣ \vdash \hat{0}\,q_1\,X0X␣ \vdash \hat{0}X\,q_1\,0X␣ \vdash \hat{0}XX\,q_2\,X␣ \vdash$

$\hat{0}XXX\,q_2\,␣ \vdash \hat{0}XX\,q_4\,X␣ \vdash \hat{0}X\,q_4\,XX␣ \vdash \hat{0}\,q_4\,XXX␣ \vdash q_4\,\hat{0}XXX␣ \vdash$

$\hat{0}\,q_1\,XXX␣ \vdash \hat{0}X\,q_1\,XX␣ \vdash \hat{0}XX\,q_1\,X␣ \vdash \hat{0}XXX\,q_1\,␣ \vdash \hat{0}XXX\,q_5\,␣$

Arbeitsweise einer TM: Läufe

Für Eingabewort w beginnt die TM mit der Startkonfiguration $q_0\, w$.

Ein Lauf ist eine maximale Folge von Konfigurationen, die durch die Übergangsrelation in Beziehung stehen

- Ein Lauf kann endlich sein, wenn es für die Schlusskonfiguration keinen Nachfolger gibt
- Ein Lauf kann unendlich sein, wenn immer neue Konfigurationen erreichbar sind

Die TM akzeptiert die Eingabe, wenn der (eindeutig bestimmte) Lauf, der mit $q_0\, w$ beginnt, endlich ist und seine letzte Konfiguration einen Endzustand beinhaltet.

Andernfalls verwirft die TM die Eingabe.

Sprache einer TM

Die Sprache einer TM wird wie erwartet definiert:

Die von einer TM $\mathcal{M}$ erkannte Sprache $\mathbf{L}(\mathcal{M})$ ist die Menge aller Wörter, die von einer TM akzeptiert werden.

Zwei Gründe für Nichtakzeptanz von Wörtern:

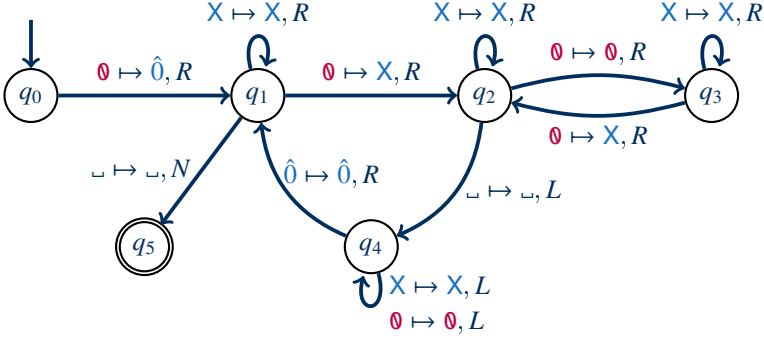
(1) TM hält in einem Zustand, der kein Endzustand ist

(2) TM hält nicht (Endlosschleife)

Es ist praktisch, wenn eine TM garantiert hält, da man Fall (2) meist nicht sicher erkennen kann (man weiß nicht, ob die TM irgendwann doch noch anhält)

Eine TM ist ein Entscheider, wenn sie bei jeder Eingabe hält. Wir sagen in diesem Fall, dass die TM die von ihr erkannte Sprache entscheidet.

Beispiel (4)



Diese TM entscheidet die Sprache $\{0^{2^i} \mid i \geq 0\}$.

Turingmaschinen programmieren

Beobachtung: Die detaillierte Beschreibung von TMs ist in der Regel sehr aufwändig

Abhilfe: Oft reicht die skizzenhafte Beschreibung der Arbeitsweise

Beispiel: Eine TM zur Erkennung von $\{\mathrm{a}^i\mathrm{b}^i\mathrm{c}^i \mid i \geq 0\}$ arbeitet wie folgt:

(1) Ersetze, angefangen von links, Vorkommen von a durch $\hat{a}$

(2) Immer wenn ein a ersetzt wurde, suche ein b und ersetze es durch $\hat{b}$, suche anschließend rechts davon ein c und ersetze es durch $\hat{c}$

(3) Gehe danach zurück zum ersten noch nicht ersetzten a und führe die Ersetzung (1) fort, bis alle a ersetzt worden sind

(4) Akzeptiere, falls der Inhalt des Bandes die Form $\hat{a}^*\hat{b}^*\hat{c}^*$ hat

(5) Andernfalls oder falls eine der Ersetzungen in Schritt (2) fehlschlägt, weil es zu wenige b oder c gibt, lehne die Eingabe ab

Berechnung vs. Worterkennung

Wir haben TMs als Automaten zur Worterkennung definiert:

- Eingabe wird am Anfang auf Band gegeben
- TM kann (a) anhalten und die Eingabe akzeptieren, (b) anhalten und die Eingabe ablehnen oder (c) nicht anhalten

Wie kann man TMs als allgemeines Rechenmodell verstehen?

(1) Berechnungsfragen können als Wortproblem kodiert werden.
   - Eingabewörter als Kodierung beliebiger Eingaben (z.B. als Binärdatei)
   - Berechnung einer Booleschen Funktion
     $\rightsquigarrow$ Entscheidungsproblem

(2) Alternative Definition: Der Inhalt des Bandes beim Halten der TM wird als Ausgabe interpretiert (keine Endzustände nötig).
Dann kodieren TMs partielle Funktionen $\Sigma^* \to \Gamma^*$ (partiell, da die TM nicht immer anhalten muss)

Die Church-Turing-These

Was ist mit dem intuitiven Begriff der „mechanisch berechenbaren“ Funktion gemeint?

- Gödel/Herbrand (1934): allgemeine rekursive Funktionen
- Church (1936): λ-Kalkül
- Turing (1936): Turingmaschine (ursprünglich „a-machine“)
- Kleene/Rosser/Church/Turing: Die drei Ansätze beschreiben die gleiche Klasse von Funktionen!

Church-Turing-These: Eine Funktion ist genau dann im intuitiven Sinne berechenbar, wenn es eine Turingmaschine gibt, die für jede mögliche Eingabe den Wert der Funktion auf das Band schreibt und anschließend hält.

- **Lesart 1:** Vorschlag einer mathematischen Definition der intuitiven Idee von Berechenbarkeit
- **Lesart 2:** „Naturgesetz“ über die Möglichkeiten und Grenzen des Rechnens an sich

Computer sind kompliziert

Das Modell der Turingmaschine ist einfach
das Verhalten von Turingmaschinen ist es nicht!

Betrachten wir einen ganz einfachen Sonderfall:

- TMs mit nur fünf Zuständen
- und nur zwei Zeichen im Bandalphabet
- bei Eingabe ϵ (leeres Wort)

Bereits diese „Spielzeugbeispiele“ werden auch 80 Jahre nach Einführung der TM noch nicht völlig verstanden:

- Es gibt TMs dieser Art, die bei Eingabe ϵ erst nach über 47 Millionen Übergängen[1] anhalten
- Es gibt TMs dieser Art, von denen wir bis heute nicht wissen, ob sie bei Eingabe ϵ jemals anhalten oder nicht

[1] diese Zahl ist für ein TM-Modell, das am linken und am rechten Bandende Speicher allozieren kann („zweiseitig unendliches Band“); siehe „Busy Beaver“.

Varianten von Turingmaschinen

Es gibt sehr viele alternative Definitionen von Turingmaschinen

- Alternative Akzeptanzbedinungen (Ausgabe der Antwort auf Band, totale Übergangsfunktion + explizite Stoppzustände für Akzeptanz & Ablehnung, ...)
- Alternative Bewegungsregeln (keine N-Übergänge, anderes Verhalten am Rand des Bandes, ...)
- TMs mit beidseitig unendlichem Band
- TMs mit mehreren Bändern
- nichtdeterministische Turingmaschinen
- TMs mit wahlfreiem Speicherzugriff (RAM)
- ...

All diese Varianten können die selben Funktionen berechnen – wenn auch zum Teil mit unterschiedlichem Aufwand

$\rightsquigarrow$ Untermauerung der Church-Turing-These

TMs mit mehreren Bändern

Die Mehrband-Turingmaschine

- ... verwendet eine vorher festgelegte Zahl $k \geq 2$ von (einseitig unendlichen) Speicherbändern
- ... erweitert die Übergangsfunktion entsprechend:

$$Q \times \Gamma^k \to Q \times (\Gamma \times \{L, R, N\})^k$$

- ... erhält die Eingabe auf dem ersten Band, während die anderen anfänglich leer sind

Wichtig: Die TM hat einen unabhängigen Lese-/Schreibkopf für jedes Band (unterschiedliche Bewegungen/Positionen möglich)

Beispiel: Ein deterministischer PDA kann leicht durch eine 2-Band-TM simuliert werden. Dabei wird vom ersten Band nur gelesen, während auf dem zweiten Band der aktuelle Kellerinhalt gespeichert wird.

Mehr Bänder ≠ mehr Ausdrucksstärke

Jede Mehrband-TM ist äquivalent zu einer TM (mit einem Band).

Beweis: Es ist klar, dass eine TM durch eine Mehrband-TM simuliert werden kann, indem einfach nur ein Band genutzt wird.

Umgekehrt können mehrere Bänder auf einem simuliert werden:

- Für jedes der k Bänder gibt es je ein endliches Wort (Bandinhalt) und eine Position (Lese-/Schreibkopf)
- Speicherung auf einem Band:
  - Bandinhalte werden hintereinander gespeichert, getrennt durch ein Sonderzeichen #
  - Steht der Kopf über einer Zelle mit Symbol s, dann wird dort stattdessen ein markiertes Symbol $\hat{s}$ gespeichert
- Die kodierte Startkonfiguration der k-Band-TM bei Eingabe $\mathtt{a}_1 \cdots \mathtt{a}_n$ ist also: $\#\hat{a}_1\,\mathtt{a}_2 \cdots \mathtt{a}_n\#\hat{\sqcup}\#\cdots\#\hat{\sqcup}\#$

Mehr Bänder ≠ mehr Ausdrucksstärke (2)

Jede Mehrband-TM ist äquivalent zu einer TM (mit einem Band).

Beweis (Fortsetzung): Mit dieser Kodierung kann die TM einzelne Schritte der Mehrband-TM simulieren:

- Intitialisierung: Die Eingabe wird in die Kodierung der k-Band-Konfiguration umgeschrieben
- Berechnungsschritt: Die TM läuft über das gesamte Band und liest die Symbole an den Kopf-Positionen; die k gelesenen Symbole werden in der Zustandsinformation kodiert; dann läuft die TM nochmals über das Band und aktualisiert alle k Kodierungen entsprechend der Übergangsfunktion
- Falls der Speicher für ein Band erweitert werden muss (Verlassen des bisher verwendeten Speichers nach rechts), verschiebt die TM alle darauf folgenden Zellen entsprechend, kehrt zurück und setzt die Simulation fort

Mehr Bänder ≠ mehr Ausdrucksstärke (3)

Jede Mehrband-TM ist äquivalent zu einer TM (mit einem Band).

Beweis (Fortsetzung): Die skizzierte Simulation benötigt viele Zustände, um alle relevanten Informationen zwischenspeichern zu können (Zustand der simulierten Maschine, gelesene Symbole unter den k Köpfen, Arbeitszustand bei Hilfsoperationen wie Speicherverschiebung, . . .).

$\rightsquigarrow$ Details umständlich, aber im Prinzip machbar. □

Komplexität?

- Die Zahl der Schritte zur Simulation eines Schrittes ist proportional zur Gesamtlänge des beschriebenen Speichers
- Der maximal beschriebene Speicher pro Band ist proportional zur Zahl der bereits berechneten Schritte
- Die Simulation von n Schritten benötigt also $O(kn^2)$ Schritte
  (Sofern n nicht kürzer ist als die Eingabe w; allgemeiner: $O(k\,n\max(n, |w|))$)

Zusammenfassung und Ausblick

Turingmaschinen (TMs) liefern ein allgemeines Modell der Berechnung

Die Church-Turing-These besagt, dass jeder Algorithmus in diesem Modell beschrieben werden kann

Zahlreiche Varianten von TMs führen zur gleichen Ausdrucksstärke, konkret z.B. Mehrband-TMs.

Offene Fragen:

- Wie funktionieren nichtdeterministische Turingmaschinen?
- Wo sind die Grenzen der Berechnung mit Turingmaschinen?
- Wie genau hängt das alles mit Sprachen vom Typ 1 und Typ 0 zusammen?