Turingmaschinen

- Definition und Arbeitsweise
- Zusammenhang mit Grammatiken

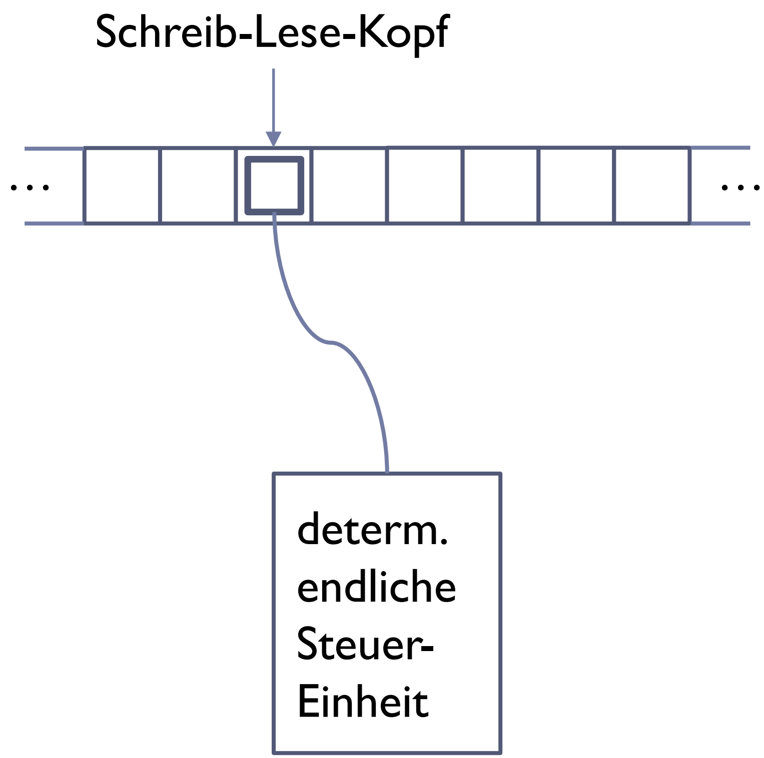
- grundlegendes Verständnis, welche Typen an Sprachen gibt es, welche Maschinen akzeptieren diese; bei allen außer Turing wissen wie die funktionieren

Chomsky-Hierarchie

Typ	Name	Erlaubte Produktionen	Akzeptierende Maschine	Beispiel
3	Regulär	N → wM $w \in T^*$	Endlicher Automat	a^n
2	Kontextfrei	N → w $w \in (N \cup T)^*$	Kellerautomat	$a^n b^n$
1	Kontext-sensitiv	uNv → uwv $u, v \in (N \cup T)^*$ $w \in (N \cup T)^+$ S → eps	Linear gebundener Automat	$a^n b^n c^n$
0	Rekursiv aufzählbar	u → v $u \in V^* N V^*, v \in V^*$ $V* = (N \cup T)^*$	Turing Maschine	Halteproblem

Skript Worsch: Seite 57-63

Definition 5.6: Turingmaschine



- Bandalphabet Y
- Blanksymbol $\square \in Y$
- Eingabealphabet $X \subseteq Y$
- Zustandsmenge Z
- Anfangszustand $z_0 \in Z$
- Menge $F_+ \subseteq Z$ akzeptierender Endzustände
- Menge $F_- \subseteq F \setminus F_+$ ablehnender Endzustände
- Überführungsfunktion $f : Z \times Y \to Z \times Y \times \{-1,0,1\}$

Definition 5.6: Turingmaschine

- Die Berechnung für Wort w beginnt so:
  - Das Band enthält w, umgeben von □ Feldern („leer"),
  - der Kopf steht auf dem ersten Symbol von w und
  - Die Steuereinheit ist in Anfangszustand z_0.

- Die Berechnung wird folgendermaßen durchgeführt:
  - $f(z, y) = (z', y', d)$ bedeutet:
    - Wenn die TM in Zustand z ist und Symbol y liest, dann
    - geht sie in Zustand z', schreibt y' und bewegt danach den Kopf um d Felder nach rechts.
  - Wenn ein akzeptierender Zustand erreicht wird, wird w akzeptiert.
  - Wenn ein ablehnender Zustand erreicht wird, wird w abgelehnt.
  - Solange keines von beidem geschehen ist, „arbeitet die TM weiter"

Definition 5.6: Turingmaschine

- „Anhalten“:
  - Wenn eine TM einen akzeptierenden oder ablehnenden Zustand erreicht hat, soll sie „anhalten“:
    - Sie soll den erreichten Zustand nicht verlassen.
    - Sie soll die Bandbeschriftung nicht mehr ändern.
    - Sie soll den Kopf nicht mehr bewegen
  - Formal:

$$\forall z \in F_+ \cup F_- \; und \; \forall y \in Y: f(z, y) = (z, y, 0)$$

- Sprache einer Turingmaschine:
  - Die von einer TM erkannte Sprache ist die Menge aller Wörter $w \in X^*$, für die die TM irgendwann einen akzeptierenden Zustand erreicht.

Beispiel: Erkennung von Palindromen

- TM zur Erkennung aller geraden Palindrome über dem Alphabet
  $X = \{a, b\}$

- Ansatz
  - Beginne am linken Wortende
  - Konsumieren den ersten Buchstaben am linken Wortende
  - Speichern den ersten Buchstaben im Zustand ab
  - Fahre ans rechte Wortende
  - Konsumiere den Buchstaben am rechten Wortende
  - Vergleiche den Buchstaben mit dem im Zustand gespeicherten
  - Erkenne Fehler oder fahre zurück ans neue linke Wortende
  - Wiederhole solange bis das ganze Wort konsumiert ist

Beispiel: Erkennung von Palindromen

- TM zur Erkennung aller Palindrome über dem Alphabet $X = \{a, b\}$

- Zustandsmenge $Z = \{r, r_a, r_b, l, l_a, l_b, f_+, f_-\}$
  - mit Anfangszustand r
  - $F_+ = \{f_+\}$
  - $F_- = \{f_-\}$

- Bandalphabet $Y = X \cup \{\square\}$

Beispiel: Erkennung von Palindromen

- TM zur Erkennung aller Palindrome über dem Alphabet $X = \{a, b\}$

- Ansatz
  - Beginne am linken Wortende (r)
  - Konsumieren den ersten Buchstaben am linken Wortende
  - Speichern den ersten Buchstaben im Zustand ab (r_a, r_b)
  - Fahre ans rechte Wortende (r_a, r_b am Ende wechsle zu l_a, l_b)
  - Konsumiere den Buchstaben am rechten Wortende
  - Vergleiche den Buchstaben mit dem im Zustand gespeicherten
  - Erkenne Fehler (f_-) oder fahre zurück ans neue linke Wortende (l)
  - Wiederhole solange bis das ganze Wort konsumiert ist (f_+)

Beispiel: Erkennung von Palindromen

- Überführungsfunktion

Alter Zustand	Gelesenes Symbol	Neuer Zustand	Neues Symbol	Kopf-Bewegung	Bemerkung
r	a	r_a	$\square$	$+1$	Symbol al linken Ende merken
r	b	r_b	$\square$	$+1$	
r	$\square$	f_+	$\square$	0	Palindrom gerade Länge erkannt
r_a	a	r_a	a	$+1$	Erstes Blanksymbol rechts der Eingabesuchen, links war a
r_a	b	r_a	b	$+1$	
r_a	$\square$	l_a	$\square$	-1	
r_b	a	r_b	a	$+1$	Erstes Blanksymbol rechts der Eingabesuchen, links war b
r_b	b	r_b	b	$+1$	
r_b	$\square$	l_b	$\square$	-1	

Beispiel: Erkennung von Palindromen

Alter Zustand	Gelesenes Symbol	Neuer Zustand	Neues Symbol	Kopf-Bewegung	Bemerkung
l_a	a	l	$\square$	-1	Symbol gleich
l_a	b	f_-	b	0	Symbol ungleich
l_a	$\square$	f_+	$\square$	0	Palindrom ungerade Länge erkannt
l_b	a	f_-	a	0	Symbol ungleich
l_b	b	l	$\square$	-1	Symbol gleich
l_b	$\square$	f_+	$\square$	0	Palindrom ungerader Länge erkannt
l	a	l	a	-1	Erstes Blanksymbol links der Eingabe suchen
l	b	l	b	-1	
l	$\square$	r	$\square$	$+1$	

Beispiel: Erkennung von Palindromen

0:	$\square$	a	b	b	b	a	$\square$
		r					
1:	$\square$	$\square$	b	b	b	a	$\square$
			r_a				
2:	$\square$	$\square$	b	b	b	a	$\square$
				r_a			
3:	$\square$	$\square$	b	b	b	a	$\square$
					r_a		
4:	$\square$	$\square$	b	b	b	a	$\square$
						r_a	
5:	$\square$	$\square$	b	b	b	a	$\square$
							r_a
6:	$\square$	$\square$	b	b	b	a	$\square$
						l_a	

Beispiel: Erkennung von Palindromen

6:	$\square$	$\square$	b	b	b	a	$\square$
						l_a	
7:	$\square$	$\square$	b	b	b	$\square$	$\square$
					l		
8:	$\square$	$\square$	b	b	b	$\square$	$\square$
				l			
9:	$\square$	$\square$	b	b	b	$\square$	$\square$
			l				
10:	$\square$	$\square$	b	b	b	$\square$	$\square$
		l					
11:	$\square$	$\square$	b	b	b	$\square$	$\square$
			r				

Beispiel: Erkennung von Palindromen

11:	□	□	b	b	b	□	□
			r				
12:	□	□	□	b	b	□	□
				r_b			
13:	□	□	□	b	b	□	□
					r_b		
14:	□	□	□	b	b	□	□
						r_b	
15:	□	□	□	b	b	□	□
					l_b		
16:	□	□	□	b	□	□	□
				l			

Beispiel: Erkennung von Palindromen

16:	$\square$	$\square$	$\square$	b	$\square$	$\square$	$\square$
				l			
17:	$\square$	$\square$	$\square$	b	$\square$	$\square$	$\square$
			l				
18:	$\square$	$\square$	$\square$	b	$\square$	$\square$	$\square$
				r			
19:	$\square$	$\square$	$\square$	$\square$	$\square$	$\square$	$\square$
					r_b		
20:	$\square$	$\square$	$\square$	$\square$	$\square$	$\square$	$\square$
				l_b			
21:	$\square$	$\square$	$\square$	$\square$	$\square$	$\square$	$\square$
				f_+			

Definition 5.7: rekursiv

Eine formale Sprache L heißt rekursiv, wenn es eine TM gibt, die

- für jede Eingabe $w \in L$ nach endlich vielen Schritten einen akzeptierenden Endzustand erreicht und
- für jede Eingabe $w \notin L$ nach endlich vielen Schritten einen ablehnenden Endzustand erreicht.

Definition 5.7: rekursiv aufzählbar

Eine formale Sprache L heißt rekursiv aufzählbar, wenn es eine TM gibt, die

- für jede Eingabe $w \in L$ nach endlich vielen Schritten einen akzeptierenden Endzustand erreicht.
- Aber für Eingaben $w \notin L$ wird nur gefordert, dass sie nicht akzeptiert werden. Die TM darf
  - irgendwann in einen ablehnenden Endzustand übergehen oder
  - unendlich arbeiten ohne je zu halten.

Beobachtung

- Wenn L rekursiv ist, dann ist L auch rekursiv aufzählbar.
- Wenn L rekursiv ist, dann ist auch $\bar{L} = X^* \setminus L$ rekursiv, also auch rekursiv aufzählbar.
- **Kurz**:

$$L\ rek. \Rightarrow\ L\ rek.aufz.und\ \bar{L}\ rek.aufz.$$

- **Mitteilung**: Die Umkehrung gilt auch:

$$L\ rek.aufz.und\ \bar{L}\ rek.aufz \Rightarrow L\ rek.$$

- **Beachte**:
- Es gibt rekursiv aufzählbare Sprachen, deren Komplement nicht rekursiv aufzählbar ist. Solche Sprachen sind also nicht rekursiv!

Definition Typ-0 Grammatik

- Eine **(erzeugende) Grammatik** ist ein Tupel $G = (N, T, S, P)$:
  - N Alphabet der **Nichtterminalsymbole**
  - T Alphabet der **Terminalsymbole**
  - $S \in N$ ausgezeichnetes **Startsymbol** und
  - $P \subset V^*NV^* \times V^*$ endliche Menge von **Produktionen**. Wobei $V = N \cup T$.
- Im Unterschied zu Typ-1 sind verkürzende Produktionen erlaubt

Satz 5.9: Typ-0-Grammatiken und rekursive Aufzählbarkeit

Eine formale Sprache kann genau dann von einer Typ-0-Grammatik erzeugt werden, wenn sie rekursiv aufzählbar ist.

Beweisidee

- Wenn w von T0G erzeugt wird, kann man das algorithmisch, also z. B. mit einer TM feststellen:
  - erzeuge erst alle Ableitungsfolgen der Länge 1,
  - dann alle Ableitungsfolgen der Länge 2,
  - dann alle Ableitungsfolgen der Länge 3,
  - usw.
- Irgendwann wird bei einer Ableitung am Ende w erzeugt.
- **Beachte**: Wenn w nicht von der Grammatik erzeugt wird, dann werden ewig Ableitungsfolgen erzeugt, die alle nicht zu w führen.
- Bei kontextsensitiven Grammatiken terminiert dieser Algorithmus garantiert

Rekursiv vs Rekursiv aufzählbar

	Turing Maschine hält im Erfolgsfall	**Turing Maschine Hält im Misserfolgsfall**	**Beispiel**
Rekursiv/ entscheidbar	Ja	Ja	
Rekursiv aufzählbar/ nicht entscheidbar	Ja	Nein	Halteproblem
Nicht rekursiv aufzählbar/ nicht entscheidbar	Nein	Nein	Äquivalenzproblem

Entscheidbarkeit vs Berechenbarkeit

- Nicht entscheidbar heisst, dass ich keinen Algorithmus schreiben kann, der garantiert mit dem richtigen Ergebnis anhält

- Nicht berechenbar heisst dass ich keinen Algorithmus formulieren kann, der das Problem löst.
  Beispiel: Busy-Beaver

Mögliche Klausuraufgaben

- Fragen zur Chomsky-Hierarchie in beliebiger Richtung (z.B. was für einen Automaten braucht man um $a^n b^n c^n$ zu akzeptieren?)

- Abgeschlossenheit von Sprachentypen bei Mengenoperationen (z.B. ist das Komplement einer rekursiven aufzählbaren Sprache auch rekursiv aufzählbar?)

L1 = {a^n}
L2 = {b^n}

L1 U L2 = {a^n, n^n } => eps, a,aa,aaa,aaaa,…,b,bb,bbb,bbbb,…

L1L2 (concat) = {a^n b^n} => abab, aabbaabb, ... immer etwas aus der einen Menge mit etwas aus der anderen Menge concatten

ODER: L3={a} L4={b} => L3 U L4={a,b} => L3L4={ab}

Ausblick: was kommt noch?

- Chomsky-Normalform und CYK-Algorithmus
  Ansatz für deterministische Syntaxanalyse

- Pumping Lemmas
  Formale Bestimmung des Sprachtyps