Info4 Stoff

Aufgabentypen:

- Grammatik → CH einordnen → NFA → DFA
- Grammatik → Chomsky-NF → CYK-Algorithmus: Tabelle / Ableitungsbäume
- Grammatik → streng kf. Grammatik
- Grammatik → Pumping Lemma Beweis, dass Gr. nicht reg, bzw. kf.
- NFA → reg. Ausdruck (KSE-Regeln)
- ☠ Sprache → Turingmaschine
- zeige, dass eine Funktion pr. rek.
- ☠ Diagonalisierung
- ☠ Entscheidbarkeit
- ☠ Sortierverfahren → Laufzeitbestimmung → Induktionsbeweis, dass Sortierung korrekt
- Heap-Sortierung (schrittweise) mit unsortiertem Binärbaum
- AVL-Baum aufbauen (mit Rotationen)

Spickerblatt:

Grammatik G = (V Variablen, $\sum$ Terminalalphabet V∩∑=Ø, P Produktionen, S Startvariable ϵ V)

kontextsensitiv (Typ 1): für alle Regeln $u \to v$ gilt $|u| \leq |v|$.

kontextfrei (Typ 2): zusätzlich: u ist eine einzelne Variable ϵ V und $|v| \geq 1$

regulär (Typ 3): zusätzlich: v ist entweder ein einzelnes Terminalzeichen, oder ein Terminalzeichen gefolgt von einer einzelnen Variablen.

Bei allen 3 Typen ist jedoch die Produktion $S \to \varepsilon$ erlaubt (ε-Sonderregelung).

BNF: zur Beschreibung von Typ 2 Grammatiken.

Regeln: statt $A \to a, A \to b$ schreibt man $A \to a \mid b$

statt $A \to ac, A \to abc$ schreibt man $A \to a[b]c$

$A \to a\{b\}c$ b kann beliebig oft vorkommen.

Bei kontextfreien Sprachen sind die Ableitungsgraphen immer Bäume.

eindeutige, kontextfreie Grammatik: für jedes Wort $w \in L(G)$ gibt es genau eine Linksableitung. Sonst mehrdeutig! streng kf.: keine ε-Produktion

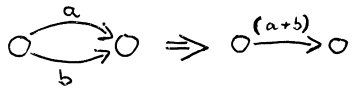
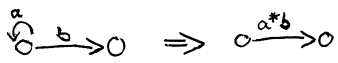
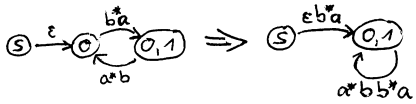
DFA M (für CH3) = (Z Zustände, $\sum$ Eingabealphabet, z_0 Startzustand, $E \subseteq Z$ Endzustände, δ Übergangsfunktionen)

NFA kann mehrere Startzustände haben. Umwandlung zu DFA immer möglich (MyHill)!

Zu jedem DFA M $\exists$ reguläre Grammatik G mit L(G) = L(M).

regulärer Ausdruck: Beispiel: Alle Worte, die 0110 enthalten. $\{(0|1)^*0110(0|1)^*\}$

Umwandlung DFA → reg. Ausdruck:

- neuen Start- und Endzustand mit ε-Übergängen
- Regel K: 
- Regel S: 
- Regel E: 

Ist die Sprache $L \subseteq \Sigma^*$ durch einen regulären Ausdruck beschreibbar $\Leftrightarrow L$ regulär.

Pumping Lemma: zeigen, dass eine Sprache L nicht regulär ist. -> uvw-Theorem:

n = Pumpingzahl, $|x| \geq n$, x =uvw, $|v| \geq 1$, $|uv| \leq n$, $uv^i w \in L$

zeigen, dass eine Sprache L nicht kontextfrei ist: uvwxy-Theorem:

$|x| \geq n$, $|vx| \geq 1$, $|vwx| \leq n$, $uv^i wx^i y \in L \quad ; i \geq 0$

Abgeschlossenheit: Die Menge L von Sprachen ist abgeschlossen unter der Operation $\circ$, falls gilt: $L_1, L_2 \in L \Rightarrow L_1 \circ L_2 \in L$.

zu jeder kontextfreien Grammatik G mit $\varepsilon \in L(G)$ gibt es eine äquivalente Grammatik G', die ε -frei ist. (äquivalent: L(G) = L(G'))

Chomsky-Normalform (CNF): Alle ε -Produktionen und alle Zyklen müssen weg! Am Schluss dürfen nur noch Produktionen der Form A→BC oder A→a existieren. Ableitungsbäume einer kontextfreien Grammatik in CNF sind immer Binärbäume, also kann ein Wort der Länge n in genau 2n-1 Schritten abgeleitet werden!

zu jeder kontextfreien Grammatik G mit $\varepsilon \notin L(G)$ gibt es eine äquivalente Gr. G' in CNF.

CYK-Algorithmus: kein exponentieller Aufwand mehr, sondern $O(n^3)$. Verfahren: zuerst umformen in CNF. Um zu prüfen, ob Wort x in L → Tabelle

Kellerautomat M (für CH2) = (Z Zustände, $\sum$ Eingabealphabet, Γ Kelleralphabet, δ Übergangsfunktionen, z_0 Startzustand, $\#\in \Gamma$ Kellerbodenzeichen). Übergang z.B. $\delta(z_0, a, \#) \rightarrow (z_0, A\#), (z_1, A\#)$

bei deterministischem KA zusätzlich $E \subseteq Z$ Endzustände Z, außerdem mus gelten:

$\forall z \in Z, \forall a \in \Sigma, \forall A \in \Gamma : |\delta(z, a, A)| + |\delta(z, \varepsilon, A| \leq 1$

deterministisch: Automat hat zu jedem Zeitpunkt höchstens eine Alternative!

eine kontextfreie Grammatik G heißt deterministisch kontextfrei, falls es einen deterministischen KA M gibt mit L(G) = L(M).

Ch2-Grammaktik G=(V, Σ, P, S) → nichtdet. KA M = ({z}, Σ , $V \cup \Sigma$, δ , z, S) :

- Für jede Regel aus P: $\delta(z, \varepsilon, S) \rightarrow (z, a)$
- zusätzlich für alle $a \in \Sigma$: $\delta(z, a, a) \rightarrow (z, \varepsilon)$

nichtdet. KA M = (Z, Σ , Γ, δ , z_0 , #) → Ch2-Grammatik G=(V, Σ , P, S) :

- V := $\{S\} \cup Z \times \Gamma \times Z$
- S → [z_0, #, z] für alle z ∈ Z
- [z, A, z'] → a für alle $\delta(z, a, A) \rightarrow (z', \varepsilon)$
- [z, A, y] → a[z', B, y] für alle $\delta(z, a, A) \rightarrow (z', B)$ und alle $y \in Z$
- [z, A, y'] → a[z', B, y][y, C, y'] für alle $\delta(z, a, A) \rightarrow (z', BC)$ und alle $y, y' \in Z$

LR(k)-Grammatik: Eine kontextfreie Grammatik ist eine LR(k)-Grammatik, wenn man durch Lookaheads der Länge k erreichen kann, dass bei einer Reduktion von Links nach rechts in jedem Schritt genau eine Regel anwendbar ist.

Jede kontextfreie Sprache, für die es eine LR(k)-Grammatik gibt, ist det. k-frei.,

Turingmaschine (für CH1 und CH0) = (Z Zustände, $\sum$ Eingabealphabet, Γ Bandalphabet, δ Übergangsfunktionen, z_0 Startzustand, $\square \in \Gamma \setminus \Sigma$ Leerzeichen, $E \subseteq Z$ Endzustände).

Übergang $\delta(z, a) \rightarrow \delta(z', b, x)$; d.h. wenn sich *M* im Zustand z befindet und unter dem Schreib-/Lesekopf das Zeichen *a* steht → Übergang zu *z'*, schreibe anstatt des *a*'s das Zeichen *b* und bewege den Schreib-/Lesekopf in Richtung *x* (R, oder L)

Konfiguration: (α, *z*, β): $\alpha\beta$ liegt auf dem Band, Maschine befindet sich im Zustand *z*, Schreib-/Lesekopf auf erstem Zeichen von β. ($\alpha, \beta \in \Gamma^*$)

Startkonfiguration einer Turingmaschine bei Eingabe *x* : (ε, x, z_0) .

linear beschränkte Turingmaschine: Sie darf nur Positionen beschreiben, an denen zu Beginn die Eingabe *x* stand.

Die von Turingmschinen akzeptierten Sprachen sind vom Typ CH0.

Die von linear beschränkten Turingmaschinen akzeptierten Sprachen sind vom Typ CH1.

Abschlusseigenschaften:

Typ	Schnitt	Vereinigung	Komplement	Produkt	Stern
CH3	✓	✓	✓	✓	✓
det kf	/	/	✓	/	/
CH2	/	✓	/	✓	✓
CH1	✓	✓	✓	✓	✓
CH0	✓	✓	/	✓	✓

Entscheidbarkeit:

Typ	Wortproblem *ist Wort x in L(G) ?*	Leerheitsproblem *ist L(G)=Ø ?*	Äquivalenzproblem $L(G_1)=L(G_2)$?	Schnittproblem $L(G_1) \cap L(G_2)$?
CH3	✓	✓	✓	✓
det kf	✓	✓	✓	/
CH2	✓	✓	/	/
CH1	✓	/	/	/
CH0	/	/	/	/

Turing-Berechenbarkeit: Es gibt eine Turingmaschine, die für alle Eingaben $bin(n_1)\#bin(n_2)\#...\#bin(n_k)$ nach endlich vielen Schritten mit $bin(f(n_1, ..., n_k))$ auf dem Band stoppt.

Jede k-Band Turingmaschine kann man durch eine 1-Band Turingmaschine simulieren.

Loop-Programme: $x_i:=c$, $x_i:=x_j+c$, $x_i:=x_j-c$ (=0, falls $c>x_j$) sind Loop-Programme.

Beispiel: IF x=0 THEN *A* END → y :=1;
LOOP x DO y:=0 END;
LOOP y DO *A*

nicht Loop-berechenbar: Ackermann-Funktion

Ist eine Funktion While- oder Goto-berechenbar, so ist sie auch Turing-berechenbar.

Klasse der primitiv-rekursiven Funktionen (entspr. Loop):

- alle konstanten Funktionen
- Nachfolgefunktion $s(n) = n+1$
- Projektionen $f(n_1, ..., n_k)=n_j$ für alle $j \in \{1, ..., k\}$. Bsp.: $p_1^3(ad(x, y), x, y)$
- Komposition: Sei *f, g* pr.rek., so ist auch $f(g(n))$ pr.rek. $\forall n \in N_0$

- Funktion, die aus pr. rek. Funktionen entsteht. Sei $F : N_0^k \to N_0$ und *g*, *h* pr.rek, dann

$$F(n, x_2, ..., x_k) = \begin{cases} g(x_2,...x_k) & , \text{falls } n = 0 \\ h(F(n-1, x_2,..., x_k), n-1, x_2,..., x_k) & , \text{sonst} \end{cases}$$

$$\text{Bsp.: } mult\ (x,y) = \begin{cases} 0 & , \text{falls } x = 0 \\ add(mult(x-1, y), y) & , \text{sonst} \end{cases}$$

Klasse der μ-rekursiven Funktionen (entspr. While, Goto):

zusätzlich μ-Operator. Sei f_μ: $N_0^k \to N_0$, dann

$$(x_1,...,x_k) \mapsto \begin{cases} \min\{n \subset N \mid f(n, x_1,...x_k) = 0\} & , \text{falls } f(m, x_1,..., x_k) \text{ def. } \forall\, m \le n \\ (undefiniert) & , \text{sonst} \end{cases}$$

Menge $A \subseteq \Sigma^*$ heißt entscheidbar, wenn:

$$\aleph_A(w) = \begin{cases} 1, & w \in A \\ 0, & w \notin A \end{cases}$$

entspr. charakteristische Fkt.

semi-entscheidbar, wenn:

$$\aleph_A(w) = \begin{cases} 1, & w \in A \\ \text{undefiniert}, & w \notin A \end{cases}$$

für alle $w \in \Sigma^*$, semi-charakteristische Fkt.

Eine Sprache $A \subseteq \Sigma^*$ ist genau dann entscheidbar, wenn sowohl *A* als auch $\overline{A} \subset \Sigma^* \setminus A$ semi-entscheidbar sind.

rekursiv aufzählbar: Für die Sprache $A \subseteq \Sigma^*$ gibt es eine berechenbare Fkt. $f : N_0 \to \Sigma^*$, so dass *A={f(0), f(1), f(2), ...}*. Genau dann ist die Sprache auch semi-entscheidbar!

spezielles Halteproblem: Sprache $H_S = \{w \in \{0,1\}^* \mid M_w \text{ angesetzt auf } w \text{ hält}\}$

allg. Halteproblem: Sprache $H = \{w\#x \in \{0,1\}^* \mid M_w \text{ angesetzt auf } x \text{ hält}\}$; # = Trennzeichen

Halteprobleme sind nicht entscheidbar!

Reduzierbarkeit: Sei $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$. *A* heißt reduzierbar auf *B* (in Zeichen $A \le B$), falls eine totale, berechenbare Fkt $f: \Sigma^* \to \Gamma^*$ existiert. Mit: $\forall x \in \Sigma^* : x \in A \Leftrightarrow f(x) \in B$.

Heap und AVL-Bäume sind binäre Suchbäume.

Heap:

- alle inneren Knoten bis auf max. einen haben genau zwei Kinder
- alle Nachfolger haben höchstens gleich großen Schlüssel
- Knoten mit weniger als 2 Kindern befinden sich auf Level größer Tiefe
- Baum unten von links nach rechts aufgefüllt

Sortierverfahren mit Heap:

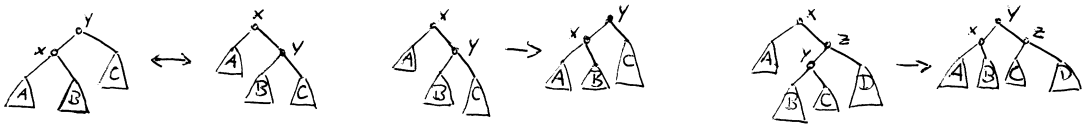
- beginne auf unterster Ebene

- falls Kind(er) größer Knoten, vertausche (größeres) Kind mit Knoten
- evtl. Wurzel weiter absenken (auch durch vertauschen)

eine Ebene nach oben

AVL-Baum: Höhe des lUB und rUB unterscheiden sich höchstens um eins.

Rechts-/Linksrotation: Einfachrotation: Doppelrotation:



evtl. Def. von (a,b)-Bäumen & Hashing (wenn noch Platz ist)