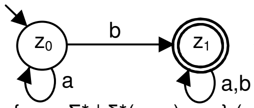
Notizen zur Theoretischen Informatik

Alphabet Σ:	endliche Menge von Symbolen, Zeichen
Zeichenkette, Wort, String:	endliche Folge von Symbolen (induktiv definierbar)
Länge $\|w\|$ eines Wortes:	Anzahl der gesamten Symbolvorkommen (rekursiv definierbar)
$\#_a(w)$:	Anzahl der Vorkommen eines Symbols in dem Wort w (rekursiv definierbar)
Präfix, Suffix:	Anfangsstück, Endstück eines Wortes (Es gibt deren n+1 bei Länge $\|w\|=n$.)
Mengenbegriffe:	Element $\in$, Teilmenge $\subseteq$, Obermenge, Durchschnitt $\cap$, Vereinigung $\cup$, Komplement co (zu vereinbarter Allmenge), Differenz \, Potenzmenge 2^M, Mächtigkeit $\|M\|$, Cantorsches Diagonalverfahren
Relationenbegriffe:	binäre Relation, aRb bzw. $(a,b)\in R$, reflexiv, symmetrisch, transitiv; Abbildungen, partielle Abbildungen; Hüllen Refl(R), Trans(R), R*; Äquivalenzrelation, Partition in Ä.-klassen
Graphenbegriffe:	binäre Relation auf/über 1 Menge, Wege als Kanten- oder Knotenfolgen (induktiv def'bar), Schleifen, Schleifenlemma, Pumping (Schleife 0- oder mehrfach durchlaufen), Verkettung von Wegen, Zwischenknotenlemma

Endlicher Automat

Alphabet Σ, Zustände Z, Anfangsszustand z_0, akzeptierende Zustände F, Übergangs- bzw. Überführungsfunktion δ: $Z \times \Sigma \rightarrow Z$ (fortsetzbar zu δ^* auf Σ^*).
Automatensprache $L(A) = \{ w \in \Sigma^* \mid \delta^*(z_0,w) \in F \}$

Bsp.:



Die $K[z] = \{ w \in \Sigma^* \mid \delta^*(z_0,w) = z \}$ (wenn z der einzige akzeptierende Zustand wäre) partitionieren Σ^*.
Die $L[z] = \{ w \in \Sigma^* \mid \delta^*(z,w) \in F \}$ (Restsprachen) partitionieren Z in Mengen äqivalenter Zustände, d.h. mit gleicher Restsprache.

Minimierungsalgorithmus Gegeben ein Automat $A=(\Sigma,Z,z_0,F,\delta)$

0-Äquivalenzklassen	– $M_{0,1}=Z\setminus F$, $M_{0,2}=F$
(k+1)-Äquivalenzklassen	– Unterteilung jeder k-Äquivalenzklasse: gleiche/ungleiche Ziel-k-Klassen nach nächstem Symbol?
Ende:	– sobald k-Äquivalenzklassen = (k+1)-Äquivalenzklassen

Der erhaltene Automat hat die minimale Anzahl von Zuständen unter allen Automaten mit der Sprache L(A).
Alle Minimalautomaten haben „die gleiche Struktur".

Zu jeder Automatensprache L=L(A) über Σ ist der **Restsprachenautomat** ein Minimalautomat. Seine Zustände sind Sprachen über Σ ! Anfangszustand(-sprache) ist L. Für einen Zustand (Sprache) H und ein $x\in\Sigma$ ist $\delta(H,x)=x^{-1}(H)$, d.h. die Menge der w mit $xw\in H$. Genau alle Zustände, die ε enthalten, sind akzeptierend, also $\in F$. Der Restsprachenautomat einer Nicht-Automatensprache ist unendlich.

Grammatik

Alphabet Σ von Symbolen (Terminalsymbolen)
Menge V von Variablen (Hilfssymbolen)
Startsymbol $S \in V$
endliche Menge von Regeln (Produktionen)
Seien beispielsweise $\Sigma = \{a,b\}$, $V = \{S,A,B\}$

Beispiele für **Regelmengen**

<u>Chomsky-Grammatik</u> (linke Seiten nicht-leer, d.h. nicht ε):
$S \rightarrow A \mid Ab$
$A \rightarrow a \mid aA$
$aAb \rightarrow ba$

<u>kontextsensitive</u> Grammatik (linke Seiten nicht länger als rechte):
$S \rightarrow A \mid Ab$
$A \rightarrow a \mid aA$
$aAb \rightarrow bab$

<u>kontextfreie</u> Grammatik (linke Seiten einzelne Variablen):
$S \rightarrow A \mid Ab$
$A \rightarrow a \mid aA$

kontextfreie Grammatik in <u>Chomsky-Normalform</u> (rechts 1 T.-Symbol / 2 Hilfssymbole)
$S \rightarrow AB \mid a$
$A \rightarrow BS \mid a$
$B \rightarrow b$

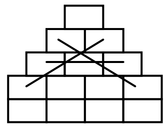
(Sonderfall Wort ε: wird von Grammatik nicht verlangt)

<u>reguläre</u> Grammatik (rechts T.-Symbol oder T.-Symbol H.-Symbol)
$S \rightarrow a \mid aA$
$A \rightarrow b \mid aS$

Grammatik G definiert Sprache L(G) (aller ableitbaren Wörter, bis auf ε):
Ableitungsbeginn: S ist ableitbar. *Ableitungsschritt*: in ableitbarem Wort eine linke Regelseite durch die rechte ersetzen.
Wort *ableitbar* : mit null, einem oder mehreren Ableitungsschritten aus S gewonnen.
L(G) besteht aus allen ableitbaren Wörtern aus Σ^* (ausschließlich aus Terminalsymbolen).

Problem: über Ableitbarkeit entscheiden — Geht prinzipiell, aber i.a. langsam, für kontextsensitive Grammatiken.

Geht schneller für kontextfreie Grammatiken in Chomsky-NF: **CYK-Algorithmus**
Für alle Teilfolgen in aufsteigender Länge Variablen suchen, aus denen sie erzeugt werden können.
Länge 1: Regeln mit Terminalsymbol rechts
Länge >1: (a) In nichtleeres Präfix+Suffix zerlegen, (b) aus deren erzeugenden Variablen Paare bilden, (c) Variablen suchen, die solche Paare auf der rechten Seite einer Regel erzeugen
Wort ist ableitbar, wenn es von S erzeugt werden kann.



ergibt Pyramidenmuster

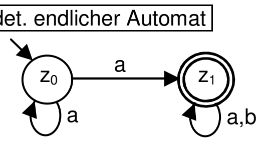
Erzeugung der **Chomsky-NF**
(1) Variablenzyklen verschmelzen. (2) In „$A\rightarrow B$" B durch alle rechten Seiten von B ersetzen. (3) In $A\rightarrow aBc$ a und c durch neue A und C ersetzen, die in neuen Regeln a bzw. c erzeugen. (4) Anstelle von $S\rightarrow ABC$ mit neuen Variablen schrittweise aufbauen, z.B. : $S\rightarrow DC$, $D\rightarrow AB$

Abschlusseigenschaften $L_1\cup L_2$, $L_1 \cap L_2$, $L_1 \setminus L_2$ und $co(L_1)$ sind (bei regulären L_1 und L_2) reguläre Sprachen

Nichtdet. endlicher Automat

Alphabet Σ, Zustände Z, Anfangsszustand z_0, akzeptierende Zustände F, Übergangs- bzw. Überführungsfunktion δ: $Z \times \Sigma \rightarrow 2^Z$ (fortgesetzt auf Σ^*).
Automatensprache $L(A) = \{ w \in \Sigma^* \mid \delta^*(z_0,w) \cap F \neq \emptyset \}$

Bsp.:



hier z.B. ist $\delta(z_0,a)=\{z_0,z_1\}$ und $\delta(z_0,b)=\varepsilon$

Berechnung eines endlichen Automaten mit derselben akzeptierten Sprache:

Zustandsmengenkonstruktion. Startmenge $\{z_0\}$. Dann immer wieder: von dort bzw. allen erreichten Zustandsmengen aus jeweils unter einem Symbol erreichbare Zustandsmengen hinzunehmen, bis keine neue mehr dazukommt.

F_{neu} besteht aus den Zustandsmengen, die Zustände aus dem F_{alt} des NDA enthalten.

Reguläre Ausdrücke

$\varnothing$ —L→ $\varnothing$	$\alpha\beta$ —L→ $L(\alpha)L(\beta)$	(Verkettung)
ε —L→ ε	$\alpha\vert\beta$ —L→ $L(\alpha)\cup L(\beta)$	(Alternative/Auswahl, oft auch als $\alpha+\beta$)
a —L→ {a}	α^* —L→ $L(\alpha)^*$	(Kleene-Hülle)

Vom Automaten (Zustände 1 bis n, anfangs 1) zum reg. Ausdruck:
Ri,j,k = Sprache der Wege von i nach j, dazwischen höchstens über 1 bis k. L(A) = Vereinigung aller R1,j,n mit $1 \le j \le n$.

Nun verwenden: Zwischenknotenlemma Ri,j,k+1 = Ri,j,k ∪ Ri,k+1,k (Rk+1,k+1,k)* Rk+1,j,k,

Pumping-Lemma (erlaubt viele Nachweise, dass Sprachen nicht regulär sind)

Es seien Σ das zugrunde liegende Alphabet und $L \subseteq \Sigma^*$ eine unendliche reguläre Sprache. Dann gibt es eine nat. Zahl n, so dass sich jedes Wort $s \in L$ mit $|s| \ge n$ in drei Wörter u, v, w aufteilen läßt,

$s = u\circ v\circ w$, so dass: (1) $|u\circ v| \le n$ (2) $|v| \ge 1$ (3) $\{ u\circ v^k\circ w \mid k \in N \} \subseteq L$ (dass s „n-aufpumpbar" ist)

Kellerautomat, Bestandteile:

Kellerautomat = Zustandsmenge Z, Anfangszustand z_0, Menge akzeptierender Zustände F, Eingabealphabet Σ, Kelleralphabet Γ, Kellerbodenzeichen \$, Transitionenmenge Δ (s.u.)

Funktionsbeschreibung:
1. Startsituation: in z_0 auf erstem Eingabe-Wort-Zeichen bei leerem Keller
2. Einzelschritte gemäß Überführungsrelation = Menge Δ von Transitionen der Art

hat 0 (falls ε), 1 oder mehrere Zeichen

(AlterZustand , SymbolAusWortOder– , KellerTopSymbolOder\$, NeuerZustand , ErsatzwortFürKellerTopSymbol)

Dabei bedeutet … – = Zeichen ignorieren, Lesekopf bleibt sitzen (sonst immer 1 Zeichen weiter).
Ersatzwort ε = KellerTopSymbol wird nur gelöscht
\$ = Boden des leeren Kellers

3. Akzeptanz: mögliche Landung in F-Zustand nach Eingabewort

Sprache des Kellerautomaten = Menge aller akzeptierten Wörter
Kellerautomatensprachen = kontextfreie Sprachen

Rezept für kontextfreie Grammatik → Kellerautomat mit gleicher Sprache:
Mit leerem Keller \$ wird das Wort ignoriert (ε) und begonnen (z0→z1, S kellern) bzw. beendet (z1→z2, welches akzeptiert).
In z1 wird
- mal ein Wortzeichen gegen dasselbe Kellerzeichen „verrechnet",
- mal eine obenauf liegende Variable im Keller durch eine passende rechte Regelseite ersetzt.

deterministischer Kellerautomat: nächster Schritt immer eindeutig
Sprachen deterministischer KA'en: <u>nicht mehr alle</u> kontextfreien Sprachen! Ein det. KA prüft aber Wörter <u>besonders schnell</u>.

Random-Access-Maschine (RAM)

... berechnet partielle Fkt. f: N → N, hat Zellen c(0), c(1), c(2), c(3), ... für je eine natürliche Zahl

Beginn: Befehlszähler b=1, Argument x in c(1), andere c(i)=0
während: bearbeitet durchnummerierte Befehle sequentiell bzw. springt/endet durch Ablaufbefehle
rechnet nur in c(0): addiert/subtrahiert Konstanten i oder Inhalte anderer Zellen c(i), subtrahiert höchstens bis herunter zu 0.
nach „End": Funktionswert f(x) in c(2) (f(x) undefiniert ←→ kein „End")

Befehlssatz:
arithmetische Befehle: Add c(i), Sub c(i), cAdd i, cSub i (i=1,2,…)
Ablaufbefehle: Goto j, If c(0) = 0 Goto j, If c(0) > 0 Goto j, End (j=1,2,…)
Umspeicher- Befehle: Load c(i), Store c(i)

k-Band-Turing-Maschine (TM)... berechnet partielle Fkt. f: N → N, hat k beidseitig unendliche Bänder aus Zellen

Zustände Z, Anfangszustand z_0, Endzustand z_e, Eingabealphabet Σ, Bandalphabet Γ (incl. B für leer und Σ), Zustandsüberführungsfunktion δ

δ(ZustandAlt, Band1Alt, ..., Band*k*Alt) =
(ZustandNeu, Band1Neu, ..., Band*k*Neu, Band1Bew, ..., Band*k*Bew)

Band*i* Alt/Neu: Zeichen in Zelle unter Lese/Schreibkopf auf Band i (Neu ersetzt Alt)
Band*i* Bew: Bewegung des Lese/Schreibkopfs auf Band i (1 Zelle R(echts)/L(inks) bzw. N(icht))
Beginn in z_0: Eingabewort w steht auf dem Ein-/Ausgabeband, LS-Kopf auf erstem Zeichen, überall sonst steht B
Ende in z_e: Ausgabewort f(w) steht auf dem Ein-/Ausgabeband, LS-Kopf auf erstem Zeichen, überall sonst steht B ← „sauberer Stil"

f(w) undefiniert ←→ z_e wird nie erreicht

Eine Sprache L heißt entscheidbar, wenn es mindestens eine Turing-Maschine M gibt, die immer
- bei Eingabe eines $w \in L$ das Ergebnis 1 und
- bei Eingabe eines $w \notin L$ das Ergebnis 0

berechnet. (M „entscheidet L")

L1 ist reduzierbar auf L2, falls man jede Turing-Maschine M2, welche die Sprache L2 entscheidet, „benutzen kann, um die Sprache L1 zu entscheiden", d.h. wenn es mindestens eine Turing-Maschine R gibt, die immer
- bei Eingabe eines $w \in L1$ ein Ergebnis $w' \in L2$ und
- bei Eingabe eines $w \notin L1$ ein Ergebnis $w' \notin L2$

berechnet.

Wenn L1 ist auf L2 reduzierbar ist, und L2 ist entscheidbar, dann ist L1 entscheidbar.
Wenn L1 ist auf L2 reduzierbar ist, und L1 ist unentscheidbar, dann ist L2 unentscheidbar.

Satz von Rice: Nicht triviale funktionale Eigenschaften von Algorithmen (z.B. „Alg. x terminiert bei Input y" (allg. Halteproblem) oder „liefert bei Input 1 den Output 1") sind nicht entscheidbar.

