

Zusammenfassung Info3

Lydia Pintscher

20. März 2005

1 Automaten

1.1 DEA

Ein deterministischer endlicher Automat besteht aus $(Q, \Sigma, \delta, s, F)$, wobei:

- Q : endliche Menge von Zuständen
- Σ : Alphabet, endliche Menge von Eingabesymbolen
- δ : Übergangsfunktion
- s : Startzustand ($\in Q$)
- F : Menge von Endzuständen ($\in Q$)

Er ist endlich, da die Zustandesmenge endlich ist und deterministisch, da δ keine Wahlmöglichkeiten lässt.

1.1.1 Sonstiges

$\epsilon \in \Sigma^*$ für alle Σ .

Jede Sprache die von einem endlichen Automaten erkannt wird ist regulär.

Eine Sprache L heißt regulär, wenn für sie einer der folgenden Punkte gilt:

1. $L = \{a\}$ mit $a \in \Sigma$
2. $L = \emptyset$
3. $L = L_1 \cdot L_2$
4. $L = L_1 \cup L_2$
5. $L = L_1^*$

wobei L_1 und L_2 reguläre Sprachen sind.

Beispiele für nicht-reguläre Sprachen:

- $a^n b a^m b a^{n+m} \mid n, m \geq 1$
- $0^{k^3} \mid k \in \mathbb{N}$

Beispiele für nicht-kontextfreie Sprachen:

- $0^i 1^j 2^i 3^j \mid i, j \geq 1$
- $a^n b^n c^n \mid n \in \mathbb{N}$

Pumping-Lemma für reguläre Sprachen: Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, sodass jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung $w = uvx$ mit $|uv| \leq n, v \neq \epsilon$, existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}$.

Pumping-Lemma für kontextfreie Sprachen: Sei L eine kontextfreie Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, sodass jedes Wort $z \in L$ mit $|z| \geq n$ eine Darstellung $z = uvwxy$ mit $|vx| \geq 1, |vwx| \leq n$, existiert, bei der auch $uv^i wx^i y \in L$ ist für alle $i \in \mathbb{N}$.

Zustände die vom Anfangszustand aus nicht erreichbar sind, heißen *überflüssig*. (Auffinden aller nicht-überflüssigen Zustände mit Tiefensuche)

Quotientensprache: $L_1/L_2 := \{w \in \Sigma^* \mid \exists z \in L_2 \text{ mit } w \cdot z \in L_1\}$

Bsp: $L_1 = \{0, 10\}$ und $L_2 = \{\epsilon, 0\}$

$L_1/L_2 = \{\epsilon, 1, 0, 10\}$

$L_1 = \{ab, b, abc\}$ und $L_2 = \{b, bc\}$

$L_1/L_2 = \{\epsilon, a\}$

Konstruktion *Äquivalenzklassenautomat* zur Minimierung:

1. trennen von Endzustände und Nichtendzuständen
2. prüfen ob die Eingabe eines Eingabesymbols Zustandsmengen trennt (zB 0), wiederholen bis alle Eingabesymbole abgearbeitet
3. nun für 2 Eingabesymbole (zB 00)
4. wiederholen bis bei einer Eingabelänge keine Zustandsmengen mehr getrennt werden

1.2 NEA

Ein nichtdeterministischer endlicher Automat besteht aus $(Q, \Sigma, \delta, s, F)$, wobei:

- Q : endliche Menge von Zuständen
- Σ : Alphabet, endliche Menge von Eingabesymbolen
- δ : Übergangsfunktion (der Automat kann sich bei Abarbeitung eines Eingabesymbols aussuchen in welchen Zustand aus Q er geht)
- s : Startzustand ($\in Q$)
- F : Menge von Endzuständen ($\in Q$)

1.2.1 Sonstiges

Zwei endliche Automaten, die die selbe Sprache akzeptieren heißen *äquivalent*.

Zu jedem NEA gibt es einen äquivalenten DEA.

Zu jedem NEA mit ϵ -Übergängen gibt es einen äquivalenten NEA ohne ϵ -Übergänge, der die selbe Sprache akzeptiert und nicht mehr Zustände hat.

Mit *Potenzmengenkonstruktion* kann ein NEA in einen DEA umgewandelt werden. Dazu Tabelle aufstellen und prüfen wohin man mit welcher Eingabe kommt. Nach und nach Tabelle auffüllen mit Zustandsmengen die dann die neuen Zustände des DEA werden.

Nerode-Relation: zur Bestimmung der Äquivalenzklassen. Eingabe unterteilen in xz , dann Fälle unterscheiden: ist das gesuchte schon in x passiert oder nicht? Dann nochmals was wenn es am Ende von x -halb erfüllt- ist. Ist der Index der Nerode-Relation von L unendlich, so ist L nicht entscheidbar.

2 Turing-Maschine

Eine deterministische Turing-Maschine besteht aus $(Q, \Sigma, \sqcup, \Gamma, s, \delta, F)$, wobei:

- Q : endliche Menge von Zuständen
- Σ : Eingabealphabet
- \sqcup : Blanksymbol (nicht in Σ)
- Γ : endliches Bandalphabet (Σ und \sqcup)
- s : Startzustand ($\in Q$)
- δ : Übergangsfunktion (mit Bewegung nach L(inks), R(echts) oder N(icht))
- F : Menge von Endzuständen ($\in Q$) (kann entfallen)

Konfiguration einer TM: $w(q)av$ bedeutet, dass sich die TM gerade im Zustand q befindet. Lesekopf steht auf Zeichen a , links davon steht w und rechts v .

Eine Nicht-deterministische Turing-Maschine hat zusätzlich eine Orakelphase. Das Orakel muss nicht die Lösungsstruktur des Problems haben. Zuerst überprüfen ob Orakel sinnvoll ist!

3 Approximationsalgorithmen

absolute Approximationsalgorithmen:

$$|OPT(I) - A(I)| \leq K, \text{ wobei } K \text{ konstant und } \in \mathbb{N}_0$$

relative Approximationsalgorithmen:

$$R_A(I) := \begin{cases} \frac{A(I)}{OPT(I)} & \text{für Minimierungsproblem} \\ \frac{OPT(I)}{A(I)} & \text{für Maximierungsproblem} \end{cases}$$

4 Problemklassen

Die Klasse NP(nichtdeterministisch polynomiell) ist die Menge aller Sprachen L , für die es eine nichtdeterministische Turing-Maschine gibt, deren Zeitkomplexitätsfunktion polynomial beschränkt ist. Alle Sprachen in NP sind entscheidbar.

Probleme die mit polynomielltem Speicherplatz gelöst werden können liegen in PSpace. Probleme die mit polypolylogarithmischem Speicherplatz gelöst werden können liegen in SC (Stevens Class).

Ein Problem A, welches mit geringem Aufwand auf ein Problem B zurückgeführt werden kann, gehört mindestens zur Komplexitätsklasse von B. B wird dann härter als A genannt. Ein Problem A ist k-hart, wenn sich alle anderen Probleme der Klasse k auf A zurückführen lassen. Liegt ein k-hartes Problem A selbst in der Klasse k, wird es k-vollständig genannt.

Wenn es einen Algorithmus gibt, der das Entscheidungsproblem in polynomieller Zeit löst so gibt es auch einen der das zugehörige Optimierungsproblem in polynomieller Zeit löst.

4.1 NP-vollständige Probleme

Ein Problem L heißt NP-vollständig genau dann, wenn L in der Klasse NP liegt und jedes Problem in NP durch eine Polynomialzeitreduktion auf L reduziert ($L_1 \leq L$) werden kann.

Um zu zeigen, dass ein Entscheidungsproblem Π NP-vollständig ist, muss man zeigen, dass Π in NP liegt (ein Orakel kann polynomieller Zeit geprüft werden). Weiterhin ist zu zeigen und dass für ein bekanntes NP-vollständiges Problem Π_2 gilt: $\Pi_2 \leq \Pi$ (indem man zeigt, dass eine Instanz des einen in eine Instanz des anderen umgewandelt werden kann).

Ein NP-vollständiges Problem heißt stark NP-vollständig, falls es auch dann noch NP-vollständig ist, wenn die Eingabe nur aus Zahlen besteht, deren Größe polynomiell in der Eingabelänge beschränkt ist. Andernfalls heißt das Problem schwach NP-vollständig. Für stark NP-vollständige Probleme kann es - unter der Annahme $NP \neq P$ - noch nicht einmal so genannte pseudopolynomielle Algorithmen geben. Das sind Algorithmen, deren Laufzeit polynomiell ist, wenn die Größe aller in der Eingabe vorkommenden Zahlen polynomiell in der Eingabelänge beschränkt sind.

Problem	Beschreibung
Traveling Salesman Problem (TSP)	Gibt es eine Tour, die alle Ecken eines vollständigen Graphen enthält und minimale Gesamtlänge unter allen solchen Touren hat? (Skript S. 49)
Erfüllbarkeitsproblem der Aussagenlogik (SAT)	Ist eine gegebene aussagenlogische Formel erfüllbar? (Skript S. 58)
3SAT	Einschränkung von SAT - Ist eine gegebene aussagenlogische Formel (in konjunktiver Normalform) mit genau 3 Literalen pro Klausel erfüllbar? (Skript S. 63)
MAX2SAT	Analog zu 3Sat - Existiert eine Wahrheitsbelegung die mindestens k Klauseln erfüllt? (Skript S. 65)
BIN PACKING Problem (BPP)	Können n Objekte in k Behälter (mit gegebener Größe) gepackt werden, ohne dass einer von ihnen überläuft?
SUBSET SUM	Gegeben sei eine Menge von Ganzzahlen $I = \{w_1, w_2, \dots, w_n\}$. Gesucht ist eine Untermenge, deren Summe gleich c ist.
COLOR	Können die Knoten eines Graphen so mit höchstens k Farben eingefärbt werden, dass zueinander benachbarte Knoten unterschiedliche Farben haben? (Skript S. 66)
3COLOR	Können die Knoten eines Graphen so mit drei Farben eingefärbt werden, dass zueinander benachbarte Knoten unterschiedliche Farben haben? (Skript S. 66)
PARTITION	Kann eine Menge natürlicher Zahlen so in zwei disjunkte Mengen aufgeteilt werden, dass deren Summe gleich ist?
CLIQUE	Gibt es in Graphen G eine Clique (Teilgraph in dem alle i, j ($i \neq j$) durch eine Kante verbunden sind) der Größe mindestens k? (Skript S. 64)
EXACT COVER	Geg: eine Menge X und Teilmengen von X. Existiert eine Menge der Teilmengen, so dass jedes Element aus X in genau einer dieser ausgewählten Teilmengen liegt? (Skript S. 69)
KNAPSACK	Skript S. 72

4.2 NP-schwere Probleme

L1 heißt NP-schwer bzw. NP-hart (von englisch NP-hard), falls alle L aus NP polynomial reduzierbar auf L1 sind. Ein Problem das NP-schwer ist muss nicht notwendigerweise in NP liegen. Man nennt ein Problem NP-schwer, wenn es mindestens so schwer ist wie alle NP-vollständigen Probleme. Ein NP-vollständiges Problem ist auch NP-schwer.

Falls L NP-schwer ist so ist auch L^c NP-schwer. Klasse NP-schwer ist also bezgl. Komplementbildung abgeschlossen. Ebenso P.

Problem	Beschreibung
SUMSET SUM	Gegeben sei eine Menge von Ganzzahlen $I = \{w_1, w_2, \dots, w_n\}$. Gesucht ist eine Untermenge, deren Summe maximal, aber nicht größer als c ist. (vgl. Subset Sum in NP-Schwer)
INTEGER PRO-GRAMMING	??? Skript S. 78

4.3 Sonstige

Postisches Korrespondenzproblem: Wortpaare sind gegeben $((a,b),(c,d),\dots)$. Kann ich sie so zusammenfügen, dass $ac\dots = bd\dots$?

5 Chomsky-Hierarchie

Grammatik $G = (\Sigma, V, S, R)$ mit:

- *Sigma*: endliches Alphabet (Terminale)
- V: endliche Menge Variablen, die nicht in Σ enthalten sind (Nichtterminale)
- S: Startsymbol aus V
- R: Ableitungsregeln/Produktionen

5.1 Chomsky-Typen

- Typ 0: semi-entscheidbar, Grammatiken ohne weitere Einschränkungen
- Typ 1: kontextsensitiv, erkannt von linear beschränkten Turingmaschinen, Regeln der Form
 - $S \rightarrow \epsilon$ oder
 - $u \rightarrow v$ mit $u \in V^+, v \in ((V \cup \Sigma) \setminus \{S\})^+$ und $|u| \leq |v|$
- Typ 2: kontextfrei, erkannt von nichtdeterministischen Kellerautomaten, Regeln der Form

- $A \rightarrow v$ mit $A \in V$ und $v \in (V \cup \Sigma)^*$
- Typ 3: regulär, rechtlinear, erkannt von endlichen Automaten, Regeln der Form
 - $A \rightarrow v$ mit $A \in V$ und $v = \epsilon$ oder $v = aB$ mit $a \in \Sigma, B \in V$

5.2 Sonstiges

Chomsky-Normalform: alle Regeln haben die Form

- $A \rightarrow BC$ oder
- $A \rightarrow a$

Es wird also kein ϵ erzeugt.

Für kontextfreie Sprachen können noch die Regeln

- $S' \rightarrow \epsilon$ und
- $S' \rightarrow S$

hinzugefügt werden. Dabei ist S' das neue Startsymbol.

Um eine Grammatik in Chomsky-Normalform zu bringen muss man:

1. Terminale durch Variablen ersetzen und neue Regel einführen Variable \rightarrow Terminal
2. Umformen von Regeln länger 3
3. ϵ -Regeln entfernen
4. Kettenregeln entfernen

Greibach-Normalform: kontextfreie Grammatik, alle Ableitungsregeln haben die Form:

- $A \rightarrow a\alpha$ mit $A \in V, a \in \Sigma$ und $\alpha \in V^*$

6 Kellerautomat

Ein (nichtdeterministischer) Kellerautomat ((N)PDA) besteht aus $(Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$, wobei:

- Q : endliche Zustandmenge
- Σ : endliches Eingabealphabet
- Γ : endliches Stack-Alphabet
- q_0 : Anfangszustand (aus Q)
- Z_0 : Initialisierung des Stack (aus Γ)
- δ : Übergangsfunktion
- F : Menge der akzeptierenden Endzustände, kann entfallen

7 Entscheidbarkeit

Eine Sprache ist entscheidbar, wenn es eine Funktion gibt die gleich 1 ist, wenn ein Wort in der Sprache enthalten ist und 0 sonst.

Eine Sprache ist semi-entscheidbar, wenn es eine Funktion gibt die gleich 1 ist, wenn ein Wort in der Sprache enthalten ist und undefiniert sonst.

Eine Sprache ist unentscheidbar, wenn diese Funktion vollständig undefiniert ist.

Um zu zeigen, dass eine Sprache entscheidbar ist, muss man nur eine Turingmaschine angeben die diese Sprache entscheidet.

Um zu zeigen, dass eine Sprache nicht entscheidbar ist, muss man sie auf eine andere nicht-entscheidbare Sprache reduzieren.

Eine Funktion heißt berechenbar, wenn es einen Algorithmus gibt, der nach endlich vielen Schritten diese Funktion berechnet und sonst nicht terminiert.

Problem	regulär	kontextfrei	kontextsens.	semientsch.	entsch.
$w \in L(G)$	e	e	e	u	e
$L(G) = \emptyset$	e	e	u	u	u
$L(G)$ endlich	e	e	u	u	u
$L(G_1) = L(G_2)$	e	u	u	u	u
$L(G_1) \subseteq L(G_2)$	e	u	u	u	u

Abgeschlossenheit	$L_1 * L_2$	L^*	$L_1 \cup L_2$	$L_1 \cap L_2$	L^c
CH-3 regulär	ja	ja	ja	ja	ja
CH-2 kontextfrei	ja	ja	ja	nein	nein
CH-1 kontextsensitiv	ja	ja	ja	ja	ja
CH-0 semi-entscheidbar	ja	ja	ja	ja	nein
entscheidbar	ja	ja	ja	ja	ja