Endliche Automaten

Ein *deterministischer endlicher Automat* besteht aus endlichen Mengen von Zuständen, Eingabesymbolen und einer Übergangsfunktion.
Er entscheidet ob eine endliche Eingabe gültig ist.

Reguläre Ausdrücke

Reguläre Ausdrücke beschreiben *reguläre Sprachen*.
Dies sind genau die Sprachen, die nach dem *Satz von Kleene* von einem DEA aktzeptiert werden.

Nichtdeterministische Automaten

Zustandsübergänge sind nichtdeterministisch.
Jeder NEA besitzt einen äquivalenten DEA.
Gebildet mit *Potenzmengenkostruktion*.

ϵ-Übergänge

Jeder NEA mit ϵ-Übergängen besitzt einen äquivalenten NEA ohne ϵ-Übergänge, der nicht mehr Zustände benötigt.

Pumping-Lemma für reguläre Sprachen

Sei L reguläre Sprache.
Dann $\exists n \in \mathbb{N} \forall w \in L : |w| > n \implies w = uvx$ mit $|uv| \leq n, v \neq \epsilon$ und $\forall i \in \mathbb{N}_0 : uv^i x \in L$.

Äquivalenzklassenautomat

Nicht erreichbare Zustände in Q sind *überflüssig*.
Diese sind in $\mathcal{O}(|Q| \cdot |\Sigma|)$ bestimmbar.
Ein Automat ohne überflüssige Zustände ist nicht zwingend minimal.
$p, q \in Q$ sind *äquivalent* ($p \equiv q$), wenn $\forall w \in \Sigma^* : \delta(p,w) \in F \iff \delta(q,w) \in F$.
Äquivalenzklassenautomat $\mathcal{A}^{\equiv}$ zu $\mathcal{A}$ ist minimal.

Turing-Maschinen

Deterministische Turing-Maschine ist def. als:

$$\mathcal{M} := (Q, \Sigma, \textvisiblespace, \Gamma, s, \delta, F)$$

Hier sind Q Zustandsmenge, Σ Eingabealphabet, $\textvisiblespace \notin \Sigma$ Blanksymbol, $\Sigma \cup \{\textvisiblespace\} \subseteq \Gamma$ Bandalphabet, $s \in Q$ Startzustand, $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$ Übergangsfunktion und $F \subseteq Q$ Endzustände.
Alle Mengen sind in einer $(D)TM$ endlich.

Church'sche These

Die Menge der Turing-berechenbaren Funktionen ist genau die Menge der im intuitiven Sinne überhaupt berechenbaren Funktionen.

Entscheidbarkeit

Eine Sprache $L \subseteq \Sigma^*$ ist *rekursiv / entscheidbar*, wenn es eine Turing-Maschine gibt, die auf allen Eingaben hält und $w \in L$ aktzeptiert gdw. $w \in L$.

Eine Sprache $L \subseteq \Sigma^*$ ist *rekursiv-aufzählbar / semi-entscheidbar*, wenn es eine TM gibt, die $w \in L$ aktzeptiert. Ihr Verhalten für $w \neq L$ ist undefiniert.

Eine Funktion $f : \Sigma^* \to \Gamma^*$ ist *(Turing)-berechenbar / totalrekursiv*, wenn es TM gibt, die für $w \in \Sigma^*$ das Wort $f(w) \in \Gamma^*$ ausgibt.

Eine Sprache $L \subseteq \Sigma^*$ ist *entscheidbar* gdw. ihre *charakteristische Funktion* berechenbar ist.

Nichtdeterministische Turing-Maschinen

Übergangsfunktion δ bietet Wahlmöglichkeiten und ϵ-Übergänge vergleichbar mit NEAs.

Orakel-Turing-Maschinen

Deterministische TM mit Orakelband zu Orakel G sowie Fragezustand q_f und Antwortzustand q_a.

Wird q_f angenommen wenn Kopf sich auf Pos. i des Orakelbandes befindet, ergibt sich Fehler, falls Wort y auf Pos. 1 bis i nicht in Σ^* ist. Sonst wird Orakelband mit $G(y)$ überschrieben, Kopf springt auf Pos. 1 zurück und Folgezustand ist q_a.

Universelle Turing-Maschinen

Sei $\mathcal{M} := (Q, \Sigma, \Gamma, \delta, s, F)$ eine Turing-Maschine.
Ihre *Gödelnummer* $\langle M \rangle$ ist definiert als:

$\mathcal{M}$ wird durch $111\text{code}_1 11\text{code}_2 11 \ldots 11\text{code}_z 111$ kodiert, code_i stellt z Funktionswerte von δ dar:

Kodiere $\delta(q_i, a_j) = (q_r, a_s, d_t)$ mit $0^i 10^j 10^r 10^s 10^t$ wobei $d_t \in \{d_1, d_2, d_3\}$ für L, R bzw. N steht.

Universelle Turing-Maschine aktzeptiert $(\langle \mathcal{M} \rangle, w)$ und simuliert $\mathcal{M}$ auf w.

Diagonalsprache

T_w ist TM mit Gödelnummer $w \in \{0,1\}^*$.
Sei $w_i \in \{0,1\}^*$ für $i = 0, 1 \ldots$.

Die *Diagonalsprache* ist definiert durch:
$L_d := \{w_i | \mathcal{M}_i \text{ aktzeptiert } w_i \text{ nicht}\}$.
L_d enthält Wörter w_i die sich als Gödelnummer interpretiert nicht selbst aktzeptieren.

L_d und L_d^c sind nicht entscheidbar.

Halteproblem

$$\mathcal{H} := \{wv | T_w \text{ hält auf Eingabe } v\}$$

$\mathcal{H}$ ist nicht entscheidbar.

Post'sches Korrespondenzproblem

Sei $K := ((x_1, y_1), \ldots, (x_n, y_n))$ endliche Folge von Wortpaaren über Σ mit $x_i, y_i \neq \epsilon$.

Gesucht ist Indizesfolge $i_1, \ldots, i_j \in \{1, \ldots, n\}$ s.d. $x_{i_1} \ldots x_{i_k} = y_{i_1} \ldots y_{i_k}$ gilt.

Das PKP ist nicht entscheidbar.

Universelle Sprache

$$L_u := \{wv | v \in L(T_w)\}$$

d.h. die Menge der Wörter wv s.d. T_w für Eingabe v hält und v aktzeptiert.
L_u ist nicht entscheidbar aber semi-entscheidbar.

Satz von Rice

Sei R die Menge aller von TM berechenbaren Funktionen und $S \subseteq R$ nicht trivial. Dann:

$$L(S) := \{\langle \mathcal{M} \rangle | \mathcal{M} \text{ berechnet Funktion aus } S\}$$

$L(s)$ ist nicht entscheidbar.

(Semi-)entscheidbare Sprachen

Entscheidbare Sprachen sind abgeschlossen unter Komplementbildung, Schnitt und Vereinigung.

Semi-entscheidbare Sprachen sind abgeschlossen unter Schnitt und Vereinigung.

Komplexitätsklassen

Sind nichtdeterministische TM wesentlich effizienter als deterministische TM? $\mathcal{P} \neq \mathcal{NP}$?

$\mathcal{NP}$-vollständige Probleme

$\mathcal{P} \subseteq \mathcal{NP}$ trivial, $\mathcal{P} \neq \mathcal{NP}$ d.h. $\mathcal{P} \subset \mathcal{NP}$ vermutet.

Eine *polynomiale Transformation* von $L_1 \subseteq \Sigma_1^*$ nach $L_2 \subseteq \Sigma_2^*$ ist $f : \Sigma_1^* \to \Sigma_2^*$ s.d. eine DTM mit polynomialer Laufzeit existiert, die f berechnet und $\forall x \in \Sigma_1^* : x \in L_1 \iff f(x) \in L_2$.
Geschrieben: $L_1 \propto L_2$.

$\mathcal{NP}$-Vollständigkeit

Eine Sprache L ist *$\mathcal{NP}$-vollständig*, wenn $L \in \mathcal{NP}$ und $\forall L' \in \mathcal{NP} : L' \propto L$.

Erfüllbarkeitsproblem (SAT)

Prüfe ob Belegungen von booleschen Variablen existiert s.d. gegebene Klauseln erfüllt werden.

SAT ist $\mathcal{NP}$-vollständig. Insb. ist *3SAT* für Klauseln mit genau drei Literalen $\mathcal{NP}$-vollständig.

Erfüllbarkeitsproblem (Max2SAT)

Prüfe ob Belegung ex. s.d. mind. K Klauseln mit jeweils genau zwei Literalen erfüllt werden.
Max2SAT ist $\mathcal{NP}$-vollständig.

Cliquen in Graphen (CLIQUE)

Prüfe ob Clique der Größe mind. K existiert.
CLIQUE ist $\mathcal{NP}$-vollständig.

Graphenfärbung (COLOR)

Prüfe ob Knotenfärbung mit max. K Farben ex.
3COLOR ist $\mathcal{NP}$-vollständig.

Mengenabdeckung (EXACT COVER)

Sei X endl. Menge und $\mathcal{S}$ Familie von Teilmengen.
Prüfe ob $\mathcal{S}' \subseteq \mathcal{S}$ ex. s.d. $\forall a \in X \exists! A \in \mathcal{S}' : a \in A$.
EXACT COVER ist $\mathcal{NP}$-vollständig.

Teilmengensumme (SUBSET SUM)

Sei M endl. Menge, $w : M \to \mathbb{N}_0$ und $K \in \mathbb{N}_0$.
Prüfe ob $M' \subseteq M$ ex. s.d. $\sum_{a \in M'} w(a) = K$.
SUBSET SUM ist $\mathcal{NP}$-vollständig.

Mengenpartitionierung (PARTITION)

Sei M endl. Menge und $w : M \to \mathbb{N}_0$.
Prüfe ob $M' \subseteq M$ ex. s.d. $\sum_{a \in M'} w(a) = \sum_{a \in M \setminus M'} w(a)$.
PARTITION ist $\mathcal{NP}$-vollständig.

Rucksackproblem (KNAPSACK)

Sei M endl. Menge, $w : M \to \mathbb{N}_0$ Gewichtsfkt., $c : M \to \mathbb{N}_0$ Kostenfkt. und $W, C \in \mathbb{N}_0$.
Prüfe ob $M' \subseteq M$ existiert s.d. $\sum_{a \in M'} w(a) \leq W$ und $\sum_{a \in M'} c(a) \geq C$.
KNAPSACK ist $\mathcal{NP}$-vollständig.

Komplementsprachen

$\mathcal{NPC}$ ist Klasse der $\mathcal{NP}$-vollständigen Sprachen.
$\mathcal{NPI} := \mathcal{NP} \setminus (\mathcal{P} \cup \mathcal{NPC})$ enthält nicht-$\mathcal{NP}$-vollständige Sprachen in $\mathcal{NP}$.
$co-\mathcal{P}$: $\Sigma^* \setminus L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{P}$.
$co-\mathcal{NP}$: $\Sigma^* \setminus L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{NP}$.

Graphenisomorphie

Prüfen von Graphen auf Isomorphie liegt in $\mathcal{NP}$ und $co-\mathcal{NP}$, ist Kandidat in $\mathcal{NPI}$ zu liegen.

Suchprobleme

Suchproblem Π ist geg. mit Menge von Beispielen D_Π und für $I \in D_\Pi$ Menge $S_\Pi(I)$ aller Lsg. von I. Die Lösung eines Suchproblems ist die Angabe von $S_\Pi(I)$ für $I \in D_\Pi$ mit $S_\Pi(I) \neq \emptyset$ falls möglich.

Beispiele sind Bestimmung einer optimalen Tour in Graph (TSP) oder eines Hamilton-Kreises.

Aufzählungsprobleme

Aufzählungsproblem Π ist geg. mit Menge von Beispielen D_Π und für $I \in D_\Pi$ Menge $S_\Pi(I)$ aller Lsg. Lösung eines Aufzählungsproblems ist $|S_\Pi(I)|$.

z.B. wie viele Hamilton-Kreise gibt es?

Turing-Reduktion

Eine *Turing-Reduktion* $\propto_T$ von Relationen $R \propto_T R'$ über Σ^* ist eine Orakel-Turing-Maschine $\mathcal{M}$ deren Orakel R' realisiert und in polynomialer Zeit die R realisierende Funktion berechnet.

$\mathcal{NP}$-Schwere

Ein Suchproblem Π ist *$\mathcal{NP}$-schwer*, wenn $\mathcal{NP}$-vollständige Sprache L ex. s.d. $L \propto_T \Pi$.
$\mathcal{NP}$-schweres Problem ist nicht zwingend in $\mathcal{NP}$.
Ein bel. Problem ist $\mathcal{NP}$-schwer, wenn es mind. so schwer ist, wie alle $\mathcal{NP}$-vollständigen Probleme.

Die Klasse der $\mathcal{NP}$-schweren Probleme ist bzgl. Komplementbildung abgeschlossen.

INTEGER PROGRAMMING

Sei $a_{ij}, b_i, c_j, b \in \mathbb{Z}_0$ für $1 \leq i \leq m$ und $1 \leq j \leq n$.
Prüfe ob $x_i, \ldots, x_n \in \mathbb{N}_0$ ex. s.d. $\sum_{j=1}^{n} c_j \cdot x_j = B$ und $\forall 1 \leq i \leq m : \sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_j$ (d.h. $A \cdot \vec{x} \leq \vec{b}$).
INTEGER PROGRAMMING ist $\mathcal{NP}$-schwer.

Raumkomplexität

Die Klasse der mit polynomialen Speicherplatz lösbaren Probleme ist $\mathcal{PSPACE}$.

Es gilt: $\mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{PSPACE}$

Pseudopolynomiale Algorithmen

Ein Algorithmus ist *pseudopolynomial*, wenn er bei Unärkodierung polynomial in Eingabelänge ist.

Für KNAPSACK gibt es einen pseudopolynomialen Lösungsalgorithmus.
Falls $\mathcal{P} \neq \mathcal{NP}$ gilt, gibt es für TSP keinen pseudopolynomialen Lösungsalgorithmus.
TSP ist *stark $\mathcal{NP}$-vollständig*.

Approximationsalgorithmen

Polynomiale Algorithmen für $\mathcal{NP}$-vollständige Optimierungsprobleme, deren Ausgabe nicht optimal aber beweisbar gut ist.

Dazu wird die Güte einer worst-case Lösung $\mathcal{A}(I)$ von Algorithmus $\mathcal{A}$ für Π mit der Optimallösung $OPT(I)$ für $I \in D_\Pi$ verglichen.

Absolute Approximationsalgorithmen

Sei Π Optimierungsproblem.
Polynomialer Algorithmus $\mathcal{A}$ ist *Approximationsalgorithmus mit Differenzengarantie* oder *absoluter Approximationsalgorithmus*, wenn für $K \in \mathbb{N}_0$ gilt:

$$\forall I \in D_\Pi : |OPT(I) - A(I)| \leq K$$

Für $\mathcal{NP}$-schweres KNAPSACK gibt es vermutlich keinen absoluten Approximationsalgorithmus.
Falls $\mathcal{P} \neq \mathcal{NP}$ gibt es für KNAPSACK, CLIQUE definitiv keinen absoluten Approximationsalgo.

Relative Approximationsalgorithmen

Sei Π Optimierungsproblem und $\mathcal{R}_\mathcal{A}$:

$$\mathcal{R}_\mathcal{A}(I) := \begin{cases} \frac{\mathcal{A}(I)}{OPT(I)} & \Pi \text{ ist Minimierungsproblem} \\ \frac{OPT(I)}{\mathcal{A}(I)} & \Pi \text{ ist Maximierungsproblem} \end{cases}$$

Polynomialer Algorithmus $\mathcal{A}$ ist *Approximationsalgo. mit relativer Gütegarantie*, falls für $K \in \mathbb{N}$ gilt:

$$\forall I \in D_\Pi : \mathcal{R}_\mathcal{A}(I) \leq K$$

$\mathcal{A}$ ist *ϵ-approximativ*, wenn:

$$\forall I \in D_\Pi : \mathcal{R}_\mathcal{A}(I) \leq 1 + \epsilon$$

Greedy-Algorithmus für KNAPSACK

Sei $w : M \to \mathbb{N}_0$ Gewichtsfunktion, $c : M \to \mathbb{N}_0$ Kostenfunktion und $W \in \mathbb{N}_0$.

Definiere Gewichtsdichten $p_i := \frac{c_i}{w_i}$.
Sortiere $p_1 \geq \cdots \geq p_n$ in $\mathcal{O}(n \log n)$.
Für $i \in \{1, \ldots, n\}$ setze $x_i := \left\lfloor \frac{W}{w_i} \right\rfloor, W := W - \left\lfloor \frac{W}{w_i} \right\rfloor \cdot w_i$

Dieser Greedy-Algorithmus $\mathcal{A}$ läuft in $\mathcal{O}(n \log n)$.

Für $\mathcal{A}$ gilt: $\forall I \in D_{\text{KNAPSACK}} : \mathcal{R}_\mathcal{A}(I) \leq 2$

Chomsky-Hierarchie

Typ 3	**Sprachklasse**	Regulär
	Rechenmodell	Endlicher Automat
	Wortproblem	entscheidbar
Typ 2	**Sprachklasse**	Kontextfrei
	Rechenmodell	ndet. Kellerautomat
	Wortproblem	entscheidbar
Typ 1	**Sprachklasse**	Kontextsensitiv
	Rechenmodell	ndet. TM in $\mathcal{NTAPE}$
	Wortproblem	entscheidbar
Typ 0	**Sprachklasse**	Rekursiv aufzählbar
	Rechenmodell	bel. Turing-Maschine
	Wortproblem	nicht entscheidbar

Es gilt $\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$.

Kontextfreie Sprachen

Die Klasse kontextfreier Sprachen ist abgeschlossen ggü. Vereinigung, Konkatenation und Kleenschem Abschluss. Sie ist nicht abgeschlossen ggü. Komplementbildung und Durchschnitt.

Für eine kontextfreie G kann in polynomialer Zeit entschieden werden, ob $L(G)$ endlich ist.

Chomsky-Normalform

Eine kontextfreie Grammatik ist in *Chomsky-Normalform*, wenn alle Regeln von Form $A \to BC$ oder $A \to a$ mit $A, B, C \in V$ und $a \in \Sigma$ sind.

Jede nicht ϵ erzeugende kontextfreie Grammatik ist in Chomsky-Normalform überführbar:

1. Regeln ergeben nur Variablen oder genau ein $a \in \Sigma$. Mit Hilfsvariablen erreichbar.
2. Alle Regeln haben Länge ≤ 2
3. Elimination von Regeln $A \to \epsilon$
4. Ersetzen von Kettenregeln (Kreise, DFS)

Cocke-Younger-Kasami Algorithmus

Entscheidet für kontextfreie Grammatik G in Chomsky-Normalform und Wort $w \in \Sigma^*$ das Wortproblem in Zeit $\mathcal{O}(|R| \cdot |w|^3)$ wobei $|R|$ Anzahl der Regeln in G ist.

Pumping-Lemma (kontextfreie Sprachen)

Für kontextfreie Sprachen L gilt: $\exists n \in \mathbb{N} \forall z \in L :$

$|z| \geq n \implies \exists u, v, w, x, y \in L : z = uvwxy$ s.d. $|vx| \geq 1, |vwx| \leq n$ und $\forall i \in \mathbb{N} : uv^i wx^i y \in L$

Greibach-Normalform

Eine kontextfreie Grammatik ist in *Greibach-Normalform*, wenn alle Regeln von Form $A \to a\alpha$ für $A \in V, a \in \Sigma$ und $\alpha \in V^*$ sind.

Jede nicht ϵ erzeugende kontextfreie Grammatik ist in Greibach-Normalform überführbar.

Für jede Grammatik G in Greibach-Normalform kann ein Kellerautomat (PDA) gebildet werden, der $L(G)$ mit leerem Stack aktzeptiert.

Unentscheidbare Probleme

Seien G, G_1, G_2 kontextfreie Grammatiken.
Die folgenden Probleme sind nicht entscheidbar:

1. Gilt $L(G_1) \cap L(G_2) = \emptyset$
2. $\forall w \in L(G)$: Syntaxbaum von w ist eindeutig
3. Ist $(L(G))^c$ kontextfrei
4. Ist $L(G_1) \cap L(G_2)$ kontextfrei
5. Gilt $L(G) = \Sigma^*$
6. Gilt $L(G_1) = L(G_2)$
7. Gilt $L_(G_1) \subseteq L(G_2)$