Inhalt – TI : Automaten und formale Sprachen SS24

(nach Modulbeschreibung)

- Formale Sprachen und verschiedene Darstellungsformen dafür, reguläre Ausdrücke, Grammatiken (Chomsky-Hierarchie, Pumping Lemmata)
- Berechnungsmodelle: endliche Automaten, Kellerautomaten,Turingmaschinen
- Ausblick auf Grenzen der Berechenbarkeit

Weiter mit Berechenbarkeit und Komplexität
in Master-LV zur theoretischen Informatik

Lernziele – TI : Automaten und formale Sprachen SS24

(nach Modulbeschreibung)

Die Studierenden sind in der Lage, wichtige Klassen formaler Sprachen als Grundlage von Programmier- und Beschreibungssprachen einzuordnen und kennen die wesentlichen Eigenschaften der Sprachklassen.

Sie kennen die entsprechenden abstrakten Maschinenmodelle und Algorithmen und können sie zur Darstellung und Lösung praktischer Aufgabenstellungen einsetzen.

Die Studierenden wissen, dass nicht jedes formal darstellbare Problem algorithmisch lösbar ist.

Formale Sprachen

- endliches Alphabet X
- Wort $w \in X^*$
  Operationen auf Wörtern: $\circ, ^*, ^R$
  Relationen auf Wörtern: Infix, Präfix, Postfix,
  lexikographische und quasilexikographische Ordnung
- (formale) Sprache $L \subseteq X^*$
  Operationen auf Sprachen: $\cup, \cap, \bar{}, \setminus, \circ, ^*, ^R$
- Reguläre Ausdrücke
  (endliche formale Darstellung unendlicher Sprachen)

Aufgaben (Probleme) als formale Sprachen

Entscheidungsproblem (Aufgabe) =
Menge (Sprache) der Codierungen aller korrekten Instanzen

Beispiel: Wortproblem für Typ-3-Grammatiken
Wortproblem $\mathrm{WP}_{\mathrm{G3}} = \{(G, w) \mid w \in L(G)\}$

Parameter (Eingaben für Lösungsverfahren) : (G, w),

- Typ-3-Grammatik $G = (N, T, P, S)$ und
- $w \in T^*$

Frage: Gilt $w \in L(G)$?

Antwort (mögliche Ausgaben des Lösungsverfahrens):

- ja (1), falls $w \in L(G)$
- nein (0), falls $w \notin L(G)$

Instanzen z.B. mit $G = (\{S\}, \{a, b\}, \{S \to aS|b\}, S)$:

- $(G, aab) \in \mathrm{WP}_{\mathrm{G3}}$ (korrekte Instanz),
- $(G, aba) \notin \mathrm{WP}_{\mathrm{G3}}$

Aufgaben und Instanzen

Aufgabe (Problem): Sprache $P \subseteq X^*$
(Codierungen einer Klasse von verwandten Aufgaben)

Instanz eines Problems: Wort $w \in X^*$
(Codierung einer speziellen Aufgabe aus dieser Klasse)

Lösungsverfahren für das Problem P:
Entscheidungsverfahren für P, d.h. Verfahren, welches für jede Instanz (Wort) $w \in X^*$ bestimmt, ob $w \in P$

Beispiel: Vorkommen einer Zahl in einer Folge natürlicher Zahlen

Aufgabe

$$P = \{((x_1, \ldots, x_n), y) \in \mathbb{N}^* \times \mathbb{N} \mid \exists k \in \{1, \ldots, n\} : x_k = y\}$$

Menge aller Paare $((x_1, \ldots, x_n), y)$ mit $y \in \{x_1, \ldots, x_n\}$

Instanz Wort $((1, 3, 4, 6), 4)$
repräsentiert die Frage : Gilt $((1, 3, 4, 6), 4) \in P$?
Antwort: Ja
Begründung: $((1, 3, 4, 6), 4) \in P$, weil $4 \in \{1, 3, 4, 6\}$
(aber $((1, 3, 4, 6), 5) \notin P$)

Lösungsverfahren (Entscheidungsverfahren):
(beliebiges) Suchverfahren in Folgen

Aufgaben und Instanzen – Beispiele

- SAT (Erfüllbarkeitsproblem der Aussagenlogik):

  Aufgabe: $\mathrm{SAT} = \{\varphi \in \mathrm{AL} \mid \mathrm{Mod}(\varphi) \neq \emptyset\}$
  Menge aller erfüllbaren aussagenlogischen Formeln

  Instanz: $\neg a \wedge (a \vee b \vee c) \wedge \neg c \wedge \neg b$
  (Gilt $(\neg a \wedge (a \vee b \vee c) \wedge \neg c \wedge \neg b) \in \mathrm{SAT}$?)
  Nein, weil $\mathrm{Mod}(\varphi) = \emptyset$

  Lösungsverfahren (Entscheidungsverfahren):
  - semantisch: Wahrheitswerttabellen
  - syntaktisch: Kalküle, z.B. Resolution, äquivalente Umformungen

- PCP (Postsches Korrespondenzproblem)

  Problem : $\mathrm{PCP} = \{((x_1, y_1), \ldots, (x_n, y_n)) \mid \exists w \in \{1, \ldots, n\}^+ : x_{w_1} \circ \cdots \circ x_{w_{|w|}} = y_{w_1} \circ \cdots \circ y_{w_{|w|}}\}$
  Menge aller lösbaren PCP-Instanzen

  Instanz : Gilt $((1, 101), (10, 00), (011, 11)) \in \mathrm{PCP}$?
  Ja, $(((1, 101), (10, 00), (011, 11)), 1321) \in \mathrm{PCP}$,
  weil $w = 1323$ eine Lösung dieser Instanz ist.

  Lösungsverfahren (Entscheidungsverfahren):
  kann nicht existieren (Beweis im TI-Master-Modul)

Aufgaben und Entscheidungsverfahren (Ausblick)

Aufgabe P (formale Sprache) ist

entscheidbar gdw. wenigstens ein Entscheidungsverfahren für P existiert
(i.A. existieren mehrere)
z.B. Wortprobleme für Typ-1,2,3-Sprachen
SAT, Suche in Folgen, Sortieren von Folgen
durch verschiedene Such- und Sortierverfahren

unentscheidbar gdw. kein Entscheidungsverfahren (TM, Programm,
Algorithmus, ...) für P existiert,
z.B. PCP, Wortproblem für Typ-0-Sprachen,
Gültigkeitsproblem für FOL

Entscheidbarkeit eines Problems (Sprache) P wird i.A. durch Angabe
eines Entscheidungsverfahrens für P gezeigt.

Unentscheidbarkeit eines Problems P wird oft durch Reduktion bekannter
unentscheidbarer Probleme (oft HP) auf P gezeigt.
(mehr dazu im TI- Master-Modul)

Wortersetzungssysteme

Wortersetzungssystem (SRS) $S \subseteq X^* \times X^*$
(Menge von Ersetzungsregeln)

Ableitungsrelation $\to_S$ zum SRS S, transitiver Abschluss $\to_S^*$

durch Wortersetzungssystem S und Menge W von Wörtern definierte Sprache

$$L(S, W) = \{v \mid \exists u \in W : u \to_S^* v\}$$

endliche formale Darstellung unendlicher Sprachen

Wortproblem für Wortersetzungssysteme:
$\{(S, u, v) \mid S \subseteq X^* \times X^* \text{ (endlich)} \wedge u \in X^* \wedge v \in X^* \wedge u \to_s^* v\}$
(Menge aller Tripel (S, u, v) mit $u \to_s^* v$)

Wortproblem (S, u, v) (= Frage: Gilt $v \in L(S, \{u\})$?)

- im Allgemeinen unentscheidbar
- für nichtverkürzende SRS entscheidbar
- für nichtverlängernde SRS entscheidbar

Grammatiken

(endliche Darstellung unendlicher Sprachen)

Grammatik $G = (N, T, P, S)$

Ableitungen in Grammatiken

von Grammatik G erzeugte Sprache $L(G)$

Wortproblem für Grammatiken:
$\{(G, v) \mid G = (N, T, P, S) \wedge v \in T^* \wedge v \in L(G)\}$
(Menge aller Paare (G, v) mit $v \in L(G)$)

Ableitungsbäume für kontextfreie Grammatiken

eindeutige / mehrdeutige kontextfreie Grammatiken

Chomsky-Hierarchie

Eine Grammatik $G = (N, T, P, S)$ ist vom Chomsky-Typ

0 immer,

1 , falls für jede Regel $(l \to r) \in P$ gilt: $|l| \leq |r|$
(monoton, kontextsensitiv)

2 , falls Typ 1 und für jede Regel $(l \to r) \in P$ gilt: $l \in N$
(kontextfrei)

3 , falls Typ 2 und für jede Regel $(l \to r) \in P$ gilt:
$l \in N$ und $r \in (T \cup TN)$
(regulär)

Sprache L hat Chomsky-Typ i gdw. eine Grammatik G mit Chomsky-Typ i mit $L(G) = L \setminus \{\varepsilon\}$ existiert

übliche Vereinbarung: Zulassung der ε-Sonderregel
Grammatik $G = (N, T, P, S)$ hat auch dann Chomsky-Typ i, wenn sie

- nur Regeln der für Chomsky-Typ i erlaubten Form
- und evtl. die Regel $S \to \varepsilon$ enthält. In diesem Fall darf S in keiner Regel in P auf der rechten Regelseite vorkommen.

Endliche Automaten

NFA $A = (X, Q, \delta, I, F)$

- akzeptierende Pfade $\in (I \times Q^* \times F) \cup (I \cap F)$
- akzeptierte Wörter
- akzeptierte Sprache $L(A) \subseteq X^*$
- Äquivalenz, Isomorphie
- Eigenschaften:
  vollständig, deterministisch, mit ε-Übergängen, Minimalautomat
- Abschluss unter $\cup, \cap, \overline{\ }, \circ, ^*, ^R$
- Nachweis Nicht-Regularität durch Schubfachschluss

NFA akzeptieren genau alle regulären Sprachen (Typ 3)
= genau alle durch reguläre Ausdrücke darstellbaren Sprachen

Wortproblem für Typ-3-Sprachen in $O(n)$ lösbar (Minimalautomat)
(n = Länge des Eingabewortes)

Anwendung z.B. in String-Matching-Algorithmen
(siehe LV Algorithmen und Datenstrukturen)

Kellerautomaten

PDA $A = (X, Q, \Gamma, \delta, q_0, F, \bot)$

- Konfiguration, Konfigurationenfolge
- akzeptierte Wörter
- akzeptierte Sprache
- Äquivalenz
- Akzeptanz durch akzeptierende Zustände

PDA akzeptieren genau die kontextfreien Sprachen (Typ 2).
(DPDA akzeptieren echte Teilmenge)
kontextfreie Sprachen:

- Chomsky-Normalform
- CYK-Algorithmus (dynamische Programmierung)
  löst Wortproblem für Typ-2-Sprachen in $O(n^3)$
- Abschluss unter $\cup, \circ, ^*, ^R$, Schnitt mit regulären Sprachen
  aber nicht unter $\cap, \overline{\ }$
- Nachweis Nicht-Kontextfreiheit durch Schubfachschluss
- Pumping-Lemmata für kontextfreie und reguläre Sprachen

Anwendung z.B. beim Parsen von Programmen und
(arithmetischen, regulären, logischen, . . .) Ausdrücken

Linear beschränkte Automaten / Turing-Maschinen

TM $M = (X, Q, \Gamma, \delta, q_0, F, \square)$

- deterministisch / nichtdeterministisch
- Konfiguration, Konfigurationenfolge
- akzeptierte Wörter
- akzeptierte Sprache $L(M) \subseteq X^*$
- Akzeptanz durch akzeptierende Zustände
- Abschluss Turing-akzeptierbarer Sprachen unter $\cup, \cap, \circ, ^*, ^R$ aber nicht unter $^-$

TM akzeptieren genau alle durch eine Grammatik erzeugten Sprachen (Chomsky-Typ 0).

LBA (TM mit linear beschränktem Band) akzeptieren genau alle kontextsensitiven Sprachen (Chomsky-Typ 1).

LBA-Problem (offen): Ist jede von einen nichtdeterministischen LBA akzeptierte Sprache durch einen deterministischen LBA akzeptierbar?

Beispiel: TM für Verschiebung

TM $M = (\{1, \bullet\}, \{q_0, q_1, q_2, f\}, \{1, \bullet, \square\}, \delta, q_0, \{f\}, \square)$ mit

$$\begin{aligned} \delta \quad = \{ \quad & (1, q_0, 1, q_0, R) \\ & (\bullet, q_0, 1, q_1, R) \\ & (1, q_1, 1, q_1, R) \\ & (\square, q_1, \square, q_2, L) \\ & (1, q_2, \square, f, L)\} \end{aligned}$$

akzeptiert die Sprache $L(M) = L(1^* \bullet 1^*)$,
verschiebt dabei den Teil nach dem $\bullet$ um eine Zelle nach links

Nebenwirkung:
Transformation des Bandinhaltes von Beginn bis Halt der TM
als Berechnung mit Unärdarstellungen natürlicher Zahlen: Addition

Ausblick: TM zur Berechnung von Funktionen

Beobachtung (der Nebenwirkung):
deterministische TM M berechnet eine Funktion $f_M : X^* \to \Gamma^*$

Eingabe $w \in X^*$: Inhalt des Arbeitsbandes bei Start der Berechnung (Eingabewort)

Ausgabe $v \in \Gamma^*$: Inhalt des Arbeitsbandes nach Halt der TM

Falls M bei Eingabe von w nicht hält, ist $f_M(w)$ nicht definiert.

TM M berechnet i.A. eine partielle Funktion

$$f_M : X^* \to \Gamma^*$$

da f_M nur für durch Halt akzeptierte Wörter $w \in X^*$ definiert ist

Maschinenmodelle

NFA, DFA, PDA, LBA und Turing-Maschinen können

- Sprachen $L \subseteq X^*$ akzeptieren.

deterministische Turing-Maschinen können außerdem

- (partielle) Funktionen $f : X^* \to X^*$ berechnen
  (Zahlen, Tupel von Zahlen,. . . codiert als Wörter in X^*, z.B. für $X = \{0, 1\}$)

Ausblick: Nicht Turing-berechenbare Funktionen

Ist jede partielle Funktion $f : X^* \to X^*$
($f : X^* \to Y^*$, $f : \mathbb{N} \to \mathbb{N}$, $f : \mathbb{N}^n \to \mathbb{N}$) Turing-berechenbar?

Nein (Gegenbeispiel später)

Begründung: (Aussage für $X = \{0, 1\}$ zu zeigen, genügt)

1. Wieviele Turing-Maschinen über einem endlichen Alphabet gibt es?
   abzählbar viele
   (weil TM endlich codiert werden können,
   nach erstem Diagonalverfahren von Cantor)
2. Wieviele partielle Funktionen $f : X^* \to X^*$ gibt es?

   überabzählbar viele

   (zweites Diagonalverfahren von Cantor)

Damit existieren sogar sehr viel mehr (überabzählbar viele) partielle Funktionen $f : X^* \to X^*$ ($f : X^* \to Y^*$, $f : \mathbb{N} \to \mathbb{N}$, $f : \mathbb{N}^n \to \mathbb{N}$), die nicht von TM berechnet werden können.

Warum sind TM als Berechnungsmodell trotzdem interessant?

Intuitiver Berechenbarkeitsbegriff

Eine partielle Funktion $f : X^* \to X^*$ ist berechenbar, falls

- eine Rechenvorschrift oder
- ein Algorithmus oder
- ein Programm (in gängiger Programmiersprache) oder
- eine deterministische Turingmaschine

existiert, welche für jedes $x \in \mathbb{N}$ bei Eingabe von x den Wert $f(x)$ ausgibt, falls $f(x)$ definiert ist.

weitere Berechnungsmodelle:
Registermaschinen, Loop-Programme, While-Programme,
partiell rekursive Funktionen, λ-Kalkül, ...

(mehr dazu in anderen LV)

These von Church

(Alonzo Church 1903-1995)

Für alle bisher vorgeschlagenen intuitiven Definitionen der Berechenbarkeit lässt sich beweisen, dass sie dieselbe Klasse von Funktionen repräsentieren.

These von Church:
Die Menge aller Turing-berechenbaren Funktionen ist genau die Menge aller intuitiv berechenbaren Funktionen.

(Diese Aussage ist sinnvoll, aber wegen des Bezugs auf den ungenauen Begriff „Intuition“ nicht beweisbar.)

Ausblick: Komplexitätstheorie

(Entscheidungs-)Probleme:

- Problem: Sprache
  (Menge aller Codierungen einer Klasse von Aufgaben)
- Instanz: Wort
  (Codierung einer speziellen Aufgabe)
- Lösungsverfahren:
  Entscheidungsverfahren für die Sprache

Aufwandsabschätzung: Laufzeit, Speicherplatz

Einteilung von Problemen (Sprachen) nach Aufwand
(der besten bekannten Algorithmen und schwierigsten Instanzen)
in Komplexitätsklassen

Ausblick: Prominente Komplexitätsklassen

P z.B.
Sortieren, Suchen, Wortproblem Typ-2,3-Sprachen ...

NP z.B.
SAT, HC, DHC, TSP, Bin Packing, Vertex Cover,
Graph-Färbbarkeit, Äquivalenz regulärer Ausdrücke,
kombinatorische Spiele (Sudoko, n-Damen, ...)
Lösen: Raten einer Lösung (ndet.) + Überprüfen (in P)

Nachweis der NP-Vollständigkeit durch polynomielle Reduktion bekannter NP-vollständiger Probleme
(oft SAT, Graphenprobleme)

PSPACE z.B.
QBF-Erfüllbarkeitsproblem,
Erfüllbarkeitsproblem für Modal-, Beschreibungs- und Zeitlogiken,
Planen, Verifikation
Planungsspiele (Sokoban, Lunar Lockout, ...)

Nachweis der PSPACE-Vollständigkeit durch polynomielle Reduktion bekannter PSPACE-vollständiger Probleme
(oft QBF)

Prüfung TI : Automaten und formale Sprachen SS24

Klausur (90 min)
am Freitag, 9.8.2024 um 8:00-9:30 Uhr
in 3 HS TR-A1.X

- bekannte Aufgabentypen
- einzige zugelassene Hilfsmittel:
  - Stift, leeres Papier
  - ein (beidseitig) handbeschriebenes A4-Blatt

  nicht zugelassen sind also insbesondere:
  Literatur, Unterlagen, eigene Notizen zum Modul,
  Ergänzungen jeglicher Form
  elektronische Geräte jeder Art
  (insbes. Taschenrechner, Smartwatch, Telefon, . . .)

Prüfungsvoraussetzungen

Zur Prüfung ist zugelassen, wer alle folgenden Bedingungen erfüllt:

Autotool: $\geq \lfloor 34/2 \rfloor = 17$ Punkte für Pflicht-Aufgaben und

Übungen: ≥ 3 Punkte für erfolgreiches Vorrechnen

im SS24 erworbene Zulassungen: 70 (+1)