Theoretische Informatik 2

Sabine Kuske
Linzer Str. 9a, OAS 3005
Tel.: 2335, 8794
kuske@informatik.uni-bremen.de
www.informatik.uni-bremen.de/theorie

7. April 2008

Unlösbarkeit des Wortproblems für Typ-0-Sprachen

Beweisidee: Reduziere das Halteproblem (HP) auf das Wortproblem (WP).

Beweisskizze: Sei $red(TM, w) = (TM', w)$, wobei TM' wie folgt arbeitet:

1. Simuliere TM mit Eingabe w.
2. Gehe in einen Endzustand.

1. red ist berechenbar.
2. $\mathrm{HP}(TM, w) = T \iff \mathrm{WP}(TM', w) = T$.

Beweis der Unlösbarkeit mittels Reduktion

Reduktion

Seien $dp_i \colon A_i \to BOOL$ $(i = 1, 2)$ Entscheidungsprobleme.

dp_1 ist auf dp_2 reduzierbar, falls es eine berechenbare Funktion $red \colon A_1^* \to A_2^*$ gibt, so dass

$$dp_1(w) = T \iff dp_2(red(w)) = T \text{ für alle } w \in A_1^*.$$

Unlösbarkeit des Halteproblems

Halteproblem

- Eingabe: Turingmaschine TM, Wort w [1)]
- Ausgabe: T, falls TM bei Eingabe w anhält
  F sonst.

Theorem

Das Halteproblem ist nicht berechenbar.

1) (TM, w) kann eindeutig als Wort kodiert werden.

Typ	Bezeichnung	Automaten	Leerheitsproblem
0	allgemein	Turing-Maschinen	-
1	kontext-sensitiv, monoton	linear beschränkte Automaten	-
2	kontextfrei	Kellerautomaten	vgl. Übungsblatt 5
3	regulär, rechtslinear	endliche Automaten	+

Durchschnittsleerheitsproblem (DLP)

- Gegeben: L, L'.
- Es ist zu entscheiden, ob $L \cap L' = \emptyset$

Komplement von DLP $(\overline{DLP})$

- Gegeben: L, L'.
- Es ist zu entscheiden, ob $L \cap L' \neq \emptyset$

Unlösbarkeit von DLP

Theorem

$\overline{DLP}$ nicht berechenbar $\Longrightarrow$ DLP nicht berechenbar.

(Verallgemeinerbar auf beliebige Entscheidbarkeitsprobleme)

Theorem

DLP ist für kontextfreie Sprachen nicht berechenbar.

Beweisidee: Reduktion des Postschen Korrepondenzproblems auf $\overline{DLP}$

Postsches Korrespondenzproblem (PCP)

- Eingabe: $(u_1, \ldots, u_n), (v_1, \ldots, v_n)$ mit $u_i, v_i \in T^*$.
- Ausgabe: T, falls $i_1 \cdots i_k$ mit $k \geq 1$ existiert,
  so dass $u_{i_1} \cdots u_{i_k} = v_{i_1} \cdots v_{i_k}$.
  F sonst.

Theorem

Das Postsche Korrespondenzproblem ist nicht berechenbar (für $\emptyset \neq T \neq \{a\}$).

Typ	Bezeichnung	Automaten	Durchschnittsleerheitsproblem
0	allgemein	Turing-Maschinen	-
1	kontextsensitiv, monoton	linear beschränkte Automaten	-
2	kontextfrei	Kellerautomaten	-
3	regulär, rechtslinear	endliche Automaten	+

Entscheidbare Sprache

Eine Sprache $L \subseteq A^*$ ist entscheidbar, falls das Wortproblem von L berechenbar ist.

Beispiel: $L(G)$, wobei $G = (N, T, P, S)$ eine monotone Grammatik ist.

Semi-entscheidbare Sprache

Eine Sprache $L \subseteq A^*$ ist semi-entscheidbar, falls die partielle Funktion $dp'_L : A^* \to BOOL$ mit

$$dp'_L(w) = \begin{cases} T, \text{ falls } w \in L \\ \text{undefiniert sonst} \end{cases}$$

berechenbar ist.

Beispiel:

$$L_{PCP} = \{ ((u_1, \ldots, u_n), (v_1, \ldots, v_n)) \mid u_i, v_i \in T^*, \\ \exists i_1, \ldots, i_k \, (k \geq 1) : u_{i_1} \cdots u_{i_k} = v_{i_1} \cdots v_{i_k} \}$$

Zusammenfassung

- Modellierung von Algorithmen mit CE-S
- Drachenkurve
- Sortieralgorithmen (insort, quicksort, mergesort, bucketsort, bubblesort)
- Ermittlung des Zeitaufwands als Zahl der Gleichungsanwendungen
- Nachweisbarkeit der dafür erforderlichen Eigenschaften

Zeitaufwand von CE-S-Spezifikationen und anderen Algorithmen

insort, quicksort, bubblesort:	$O(n^2)$
mergesort:	$O(n \log n)$
bucketsort (festes Alphabet):	$O(n)$
Drachenkurve:	$O(2^n)$
Matrizenmultiplikation (Strassen):	$O(n^{2,81})$
CYK-Algorithmus:	$O(n^3)$

Linear, quadratisch und exponentiell im Vergleich

n	n^2	2^n
10	100	$1024 \sim 10^3$
20	400	$1048576 \sim 10^6$
30	900	$1073741824 \sim 10^9$
40	1600	$10999511627776 \sim 10^{13}$
50	2500	$1125899906842624 \sim 10^{15}$
60	3600	$1152921504600686976 \sim 10^{18}$
70	4900	$1180591620717411303424 \sim 10^{21}$

ohne Gewähr

Die Klassen P und NP

Die Klasse P enthält die Probleme, die deterministisch in polynomieller Zeit lösbar sind.

NP enthält die Probleme, die nichtdeterministisch in polynomieller Zeit lösbar sind.

Es gilt: $P \subseteq NP$

Offen: $NP \subseteq P$

NP-Vollständigkeit

Ein Entscheidungsproblem dp in NP ist NP-vollständig, falls sich jedes Problem aus NP in polynomieller Zeit auf dp reduzieren lässt.

NP-vollständige Probleme

SAT, 3SAT, TSP, Rucksackproblem, . . .

Theorem

dp_0 NP-vollständig und $dp_0 \in P$ impl. $NP \subseteq P$.

Polynomieller Platzbedarf

NPSPACE:

Probleme mit nichtdeterministischen Lösungsalgorithmen, die polynomiellen Speicherplatz brauchen (im Verhältnis zur Größe der Eingabe)
Beispiel: Wortproblem für monotone Sprachen

PSPACE:

Analog für deterministische Lösungen

Es gilt: PSPACE = NPSPACE

Chomsky-Hierarchie

Typ	Bezeichnung
0	allgemein
1	monoton
2	kontextfrei
3	regulär, rechtslinear

Automatenmodelle

Typ	Bezeichnung	Automaten
0	allgemein	Turingmaschinen
1	monoton	linear beschränkte Automaten
2	kontextfrei	Kellerautomaten
3	regulär, rechtslinear	endliche Automaten

Turingmaschinen

- Nichtdeterministische Turingmaschinen erkennen dieselben Sprachen wie deterministische Turingmaschinen.
- Die von Turingmaschinen erkannten Sprachen sind genau die Typ-0-Sprachen.
- Mehrband-Turingmaschinen erkennen dieselben Sprachen wie (Einband)-Turingmaschinen.
- Turing-berechenbare Funktionen sind CE-S-berechenbar.

Churchsche These

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

Linear beschränkte Turingmaschinen

- Benutzen nur den Platz, auf dem die Eingabe steht.
- Die von linear beschränkten Turingmaschinen erkannten Sprachen sind genau die monotonen Sprachen.

Offenes Problem

Können deterministische linear beschränkte Turingmaschinen dieselben Sprachen erkennen wie nichtdeterministische linear beschränkte Turingmaschinen?

Entscheidbarkeit

- WP: Wortproblem
- LP: Leerheitsproblem
- DLP: Leerheit des Schnitts

Typ	WP	LP	DLP
0	-	-	-
1	$NPSPACE$	-	-
2	$O(n^3)$ (CKY)	+	-
3	$O(n)$	+	+

Kleines Theo-Quiz

1. Welche Sprachen lassen sich in endlicher Form beschreiben?
2. Wie kann man das machen?
3. Wofür ist das wichtig?
4. Kann man für alle endlich beschreibbaren Sprachen einen Parser konstruieren?
5. Wieviel Platz und Zeit benötigt ein Parser höchstens, um festzustellen, ob ein Wort zu einer Sprache gehört?

6. Ist das Wortproblem für Typ-0-Sprachen semi-entscheidbar?
7. Ist das Komplement des Wortproblems für Typ-0-Sprachen (semi-)entscheidbar?
8. Mit welchem Beweisverfahren wird häufig Unentscheidbarkeit gezeigt?
9. Kann man das TSP in angemessener Zeit lösen?
10. Kann man Matrizen in angemessener Zeit multiplizieren?
11. Ist das Sortierproblem in angemessener Zeit lösbar?
12. Welche Probleme kann man in angemessener Zeit lösen?

13. Welches Automatenmodell ist ebenso mächtig, wie der Computer?
14. Was ist die partielle/totale Korrektheit eines Algorithmus?
15. Ist die Frage nach der Korrektheit eines Algorithmus entscheidbar?