Complexity Theory - Notes

Ali Bektas

November 22, 2019

Contents

1 Grundlegendes

Definition 1.1. *Für eine Sprache $A \subset \Sigma^*$ ist die charakteristische Funktion $\chi_A \colon \Sigma^* \to \{0,1\}$ wie folgt definiert:*

$$\chi_A = \begin{cases} 1 & , x \in A \\ 0 & , x \notin A. \end{cases}$$

2 Grundlegende Beziehungen

Definition 2.1. *Eine monotone Funktion $f \colon \mathbb{N} \to \mathbb{N}$ heißt **echte Komplexitätsfunktion** falls es einen Transducer M gibt mit*

- $M(x) = 1^{f(|x|)}$
- $space_M(x) = O(f(|x|))$ *und*
- $time_M(x) = O(f(|x|) + |x|)$.

3 Hierarchiesätze

3.1 Unentscheidbarkeit mittels Diagonalisierung

Wir verwenden eine Kodierung , um die TMs zu kodieren. Die Kodierung einer TM M wird durch $\langle M \rangle$ bezeichnet. Wir können auch jedem Binärstring w eine TM M_w wie folgt zuordnen:

$$M_w = \begin{cases} M & , \langle M \rangle = w \\ M' & , sonst. \end{cases}$$

M' ist dabei eien beliebig aber fest gewählte TM. Für M_w schreiben wir auch M_i , wobei i die Zahl mit der Binärdarstellung $1w$ ist.

Theorem 1 (Die Unentscheidbarkeit der Diagonalsprache)**.** *: Diagonalsprache ist semi-entscheidbar aber nicht entscheidbar.*

$$D = \{\langle M_i \rangle | M_i ist\ eine\ DTM\ ,\ die\ die\ Eingabe x_i akzeptiert.\}$$

Hierbei ist

$$x_1 = \epsilon x_2 = 0, x_3 = 1, x_4 = 00, \ldots$$

die Folge aller Binärstring in lexikografischer Reihenfolge.

Angenommen , wir haben eine Tabelle wo alle semi-entscheidbaren Sprachen drin sind und die Spalten dieser Matrix aus aller Wörter in lexi. Reihenfolge *bestehen. Wenn $\bar{D}$ semi-entscheidbar wäre , würden wir sie in dieser Matrix finden also :*

$$L(M_d) = \bar{D}$$

Da diese Sprache Komplement zu der Diagonalen dieser Matrix ist ist muss es einen Eintrag geben wo es widersprüchlich zu sein scheint. Das ist die Idee.

Theorem 2. *:Für jede berechenbare Funktion $g\colon \mathbb{N} \leftarrow \mathbb{N}$ existiert eine Sprache $D_g \notin DTIME(g(n))$.*
Sei

$$D_g = \{\langle M_i\rangle | M_i ist\ eine\ DTM\ ,\ die\ die\ Eingabe x_i in \leq g(|x_i|) Schritten\ akzeptiert.\}$$

D_g ist entscheidbar : Prüfe ob x_i mittels M_i in $\leq g(|x_i|)$ Zeit entscheidbar ist.

... Unter der Annahme , dass $D_g \in DTIME(g(n))$ ist , existiert eine $g(n)$-zeitbeschränkte DTM M_d die das Komplement von D_g entscheidet.

$$L(M_d) = \bar{D}_g$$

Die Idee ist also dieselbe wie im obigen Satz.

Theorem 3 (Gap-Theorem)**.** *Es gibt eine berechenbare Funktion $g : \mathbb{N} \to \mathbb{N}$ mit*

$$DTIME(2^{g(n)}) = DTIME(g(n))$$

Wir definieren $g(n) \geq n + 2$ so , dass für $2^{g(n)}$-zeit. DTM M gilt: [1]

$$time_M(x) \leq g(|x|) für\ fast\ alle\ Eingabe x.$$

Wir definieren ein Prädikat

$$P(n,t) : t \geq n{+}2 \text{ und für } k = 1,\dots,n \text{ und alle } x \in \Sigma_k^n \text{ gilt: } time_{M_k}(x) \notin [t{+}1, 2^t].$$

Hierbei bezeichnet Σ_k das Eingabealphabet von M_k. Da für jedes n alle. Da für jedes n alle

Ob alle Wörter der Länge n durch k verschiedene Maschinen **nicht** in $[t+1, 2^t]$ Zeit erkannt werden.

$$t \geq max\{time_{M_k}(x) | 1 \leq k \leq n, x \in \Sigma_k^n, M_k(x) hält\}$$

das Prädikat P(n,t) erfüllen , können wir g(n) wie folgt induktiv definieren :

$$g(n) = \begin{cases} 2 & ,n = 0 \\ min\{t \geq g(n-1) + n | P(n,t)\} & ,n > 0 \end{cases}$$

Da P entscheidbar ist , ist g berechenbar.

[1]Warum ist dies wichtig?.

Um zu zeigen , dass jede Sprache $L \in DTIME(2^{g(n)})$ bereits in DTIME(g(n)) enthalten ist , sei M_k eine beliebige $2^{g(n)}$-zeitbeschränkte DTM mit $L(M_k) = L$. Dann muss M_k alle Eingaben x der Länge $n \geq k$ in Zeit $time_{M_k}(x) \leq g(n)$ entscheiden da andernfalls , P(n,g(n)) wegen $time_{M_k}(x) \in [g(n)+1, 2^{g(n)}]$ verletzt wäre. Folglich ist $L \in DTIME(g(n))$ da die endlich vielen Eingaben x der Länge n ¡ durch table-lookup in Zeit $n + 2 \leq g(n)$ entscheidbar sind.

3.2 Zeit- und Platzhierarchiesätze

Um D_g zu entscheiden , müssen wir einerseits die Zeitschranke $g(|x_i|)$ berechnen und andererseits $M_i(x_i)$ simulieren. Wenn wir voraussetzen dass g eine echte Komplexitätsfunktion (s. 2.1) ist , lässt sich $g(|x|)$ effizient berechnen. Für die zweite Aufgabe benötigen wir eine möglichst effiziente universelle TM.

Theorem 4 (Die Simulierung von $M_i(x_i)$ bei einer univ. TM)**.** *Es gibt eine universelle 3-DTM U , die für jede DTM M und jedes $x \in 0,1^*$ bei Eingabe $\langle M, x\rangle$ eine Simulation von M bei Eingabe in Zeit $O(|\langle M\rangle|(time_M(x))^2)$ und Platz $O(|\langle M\rangle|space_M(x))$ durchführt und dasselbe Ergebnis liefert.*

Proof. Betrachte folgende Offline-3-DTM U :

Initialisierung:U überprüft bei einer Eingabe $w\#x$ zuerst , ob w die Kodierung $\langle M\rangle$ k-DTM M ist. Falls ja , erzeugt U die Startkonfiguration K_x von M bei Eingabe x, wobei sie die Inhalte von k übereinander liegenden Feldern der Bänder von M auf ihrem 2. Band in je einem Block von kb ,
$b = \lceil log_2(||Q|| + ||\Gamma|| + 6)\rceil$, Feldern speichert und den aktuellen Zustand von M zusammen mit den gerade von M gelesenen Zeichen auf ihrem 3. Band notiert. Hierfür benötigt M' Zeit $O(kbn) = O(n^2)$

Simulation: U simuliert jeden Rechenschritt von M wie folgt: Zunächst inspiziert U die auf dem 1. Band gespeicherte Kodierung von M , um die durch den Inhalt des 3. Bands bestimmte Aktion von M zu ermitteln. Diese führt sie sodann auf dem 2. Band aus und aktualisiert dabei auf dem 3. Band den Zustand und die gelesenen Zeichen von M. Insgesamt benötigt U für die Simulation eines Rechenschrittes von M Zeit $O(kbg(n)) = O(n \cdot g(n))$ □

Corollary 4.1. *Zeithierarchiesatz*
Für jede echte Komplexitätsfunktion $g(n) \geq n + 2$ gilt:

$$DTIME(n \cdot g(n)^2) - DTIME(g(n)) \neq \emptyset$$

Proof. G.z.z : D_g ist für jede Komplexitätsfunktion 2.1 $g(n) \geq n + 2$ in Zeit $O(n \cdot g^2(n))$ entscheidbar. Betrachte folgende 4-DTM M'. M' überprüft bei einer Eingabe x der Länge n zuerst , ob x die Kodierung $\langle M\rangle$ einer k-DTM M ist. Falls ja , erzeugt M' auf dem 4. Band den String $1^{g(n)}$ in Zeit $O(g(n))$ und simuliert M(x) wie im Beweis von Theorem 4. Dabei verminder M' die Anzahl der Einsen auf dem 4. Band nach jedem simulierten Schritt von M(x) um 1. M' bricht die Simulation ab , sobald M stoppt oder der Zähler auf Band 4 den Wert

0 erreicht. M' hält genau dann im Zustand q_{ja} wenn die Simulation von M im Zustan q_{ja} endet. Nun ist leicht zu sehen , dass M' $O(n \cdot g(n)^2)$-zeitbeschränkt ist und die Sprache $D_g$entscheidet. □

Corollary 4.2.
$$P \subsetneq E \subsetneq EXPSPACE$$

Proof. Folgt unmittelbar aus dem vorigen Korollar □

Theorem 5. *Sei $f(n) \geq n+2$ eine echte Komplexitätsfunktion und gelte*

$$\limsup_{n\to\infty} \frac{g(n) \cdot logg(n)}{f(n)} = 0$$

Dann ist

$$DTIME(f(n))/DTIME(g(n)) \neq \emptyset$$

Für $g(n) = n^2$ erhalten wir beispielsweise die echten Inklusionen $DTIME(g(n)) \subsetneq DTIME(f(n))$ für die Funktionen $f(n) = n^2, n^2log^2n$.

Theorem 6. *Platzhierarchiesatz: Sind $g(n), f(n) \geq 2$ und ist f eine echte Komplexitätsfunktion mit*

$$\liminf_{n\to\infty} \frac{g(n)}{f(n)} = 0$$

dann ist

$$DSPACE(f(n))/DSPCAE(g(n)) \neq \emptyset$$

Damit lässt sich im Fall $g(n) \leq f(n)$ die Frage , ob die Inklusion von $DSPACE(g(n))$ in $DSPACE(f(n))$ echt ist , eindeutig beantworten: Sie ist genau dann echt , wenn $\liminf\limits_{n\to\infty} g(n)/f(n) = 0$ ist , da andernfalls $f(n) = O(g(n))$ ist und somit beide Klassen gleich sind.

Corollary 6.1.

$$L \subsetneq L^2 \subsetneq DCSL \subsetneq CSL \subsetneq PSPACE \subsetneq ESPACE \subsetneq EXPSPACE.$$

4 Reduktionen

4.1 Logspace-Reduktionen

Definition 4.1 (Logspacereduktion)**.** *Seien A und B Sprachen. A ist auf B **logspacereduizerbar** falls eine Funktion $f \in FL$ existiert , so dass für alle $x \in \Sigma^*$ gilt ,*

$$x \in A \iff f(x) \in B.$$