Grundlagen der Theoretischen Informatik

Komplexitätstheorie (II)

19.07.2017

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Übersicht

1. Motivation
2. Terminologie
3. Endliche Automaten und reguläre Sprachen
4. Kellerautomaten und kontextfreie Sprachen
5. Turingmaschinen und rekursiv aufzählbare Sprachen
6. Berechenbarkeit, (Un-)Entscheidbarkeit
7. Komplexitätsklassen P und NP

Komplexitätstheorie

Inhalt

- Definition der berühmten Klassen P und NP .
- Begriff der **Reduktion**: ein Problem (eine Sprache) wird auf ein zweites reduziert. Das erste Problem ist dann höchstens so schwer wie das zweite.
- Der Begriff eines NP **-schweren** Problems.
- Einige Probleme der Graphentheorie: sie sind **NP-vollständig**.
- Die wichtigsten **Komplexitätsklassen** und ihre Struktur.

Komplexitätstheorie

Welche Arten von Komplexität gibt es?

- **Zeit**
- **Speicher**

DTIME und NTIME

Definition [NTIME(T(n)), DTIME(T(n))]

Basismodell: k-DTM $\mathcal{M}$ (ein Band für die Eingabe).

Wenn $\mathcal{M}$ mit jedem Eingabewort der Länge n höchstens $T(n)$ Schritte macht, dann wird sie $T(n)$-**zeitbeschränkt** genannt.
Die von $\mathcal{M}$ akzeptierte Sprache hat **Zeitkomplexität** $T(n)$
(tatsächlich meinen wir $\max(n+1, \lceil T(n) \rceil)$).

- **DTIME**(T(n)) ist die Klasse der Sprachen, die von $T(n)$-zeitbeschränkten DTMs akzeptiert werden.
- **NTIME**(T(n)) ist die Klasse der Sprachen, die von $T(n)$-zeitbeschränkten NTMs akzeptiert werden.

DSPACE und NSPACE

Definition [NSPACE(S(n)), DSPACE(S(n))]

Basismodell: k-DTM $\mathcal{M}$, davon ein spezielles Eingabeband (**offline DTM**).

Wenn $\mathcal{M}$ für jedes Eingabewort der Länge n maximal $S(n)$ Zellen auf den Ablagebändern benutzt, dann heißt $\mathcal{M}$ $S(n)$-**speicherbeschränkt.**
Die von $\mathcal{M}$ akzeptierte Sprache hat **Speicherkomplexität** $S(n)$
(tatsächlich meinen wir $\max(1, \lceil S(n) \rceil)$)

- **DSPACE**(S(n)) ist die Klasse der Sprachen, die von $S(n)$-speicherbeschränkten DTMs akzeptiert werden.
- **NSPACE**(S(n)) ist die Klasse der Sprachen, die von $S(n)$-speicherbeschränkten NTMs akzeptiert werden.

Wieso eine *offline*-Turing-Maschine?

Bandbeschränkung von weniger als linearem Wachstum.

Wichtige Fragen

Halten, hängen nach n Schritten.

Zeitbeschränkt: Was bedeutet es, dass **eine DTM höchstens n Schritte macht**?

Halten?: Streng genommen müßte sie dann entweder **halten** oder **hängen**. Das bedeutet, im ersten Fall, dass die Eingabe **akzeptiert** wird.

Hängen?: DTM's auf beidseitig unendlichem Band können aber nicht hängen.

Stoppen nach n Schritten.

Stoppen: Wir wollen unter **eine DTM macht höchstens n Schritte** folgendes verstehen:

- Sie **hält** (und **akzeptiert** die Eingabe) innerhalb von n Schritten.
- Sie **hängt** (und **akzeptiert** die Eingabe **nicht**) innerhalb von n Schritten.
- Sie **stoppt** innerhalb von n Schritten **ohne in den Haltezustand überzugehen**. Auch hier wird die Eingabe nicht akzeptiert.

Entscheidbarkeit

Zeit: Jede Sprache aus **DTIME**(f(n)) ist entscheidbar:

Man warte einfach solange wie $f(n)$ angibt. Falls bis dahin kein "Ja" gekommen ist, ist die Antwort eben "Nein".

Speicher: Jede Sprache aus **DSPACE**(f(n)) ist entscheidbar.

Denn es gibt nur **endlich viele verschiedene Konfigurationen**. Falls also die DTM nicht terminiert (was passiert), macht man folgendes: man schreibt alle Konfigurationen auf (die komplette Rechnung). Falls die DTM nicht terminiert, muss sie in eine Schleife laufen (eine Konfiguration erreichen, die sie schon einmal hatte). Das lässt sich feststellen.

NTM/DTM

NTM vs. DTM: Trivial sind

- $\mathbf{DTIME}(f(n)) \subseteq \mathbf{NTIME}(f(n))$ und
- $\mathbf{DSPACE}(f(n)) \subseteq \mathbf{NSPACE}(f(n))$.

Versucht man aber eine NTM durch eine DTM zu simulieren, braucht man (vermutlich) exponentiell mehr Zeit.

Für die Speicherkomplexität kann man zeigen:

- $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{DSPACE}(f^2(n))$.

Zeit/Speicher

Zeit vs. Speicher: Trivial sind

- $\mathbf{DTIME}(f(n)) \subseteq \mathbf{DSPACE}(f(n))$ und
- $\mathbf{NTIME}(f(n)) \subseteq \mathbf{NSPACE}(f(n))$.

Aber $\mathbf{DSPACE}(f(n)), \mathbf{NSPACE}(f(n))$ sind viel größer.

Bandkompression/Zeitbeschleunigung

Konstante Faktoren werden ignoriert

Nur **die funktionale Wachstumsrate einer Funktion in Komplexitätsklassen zählt**: Konstante Faktoren werden ignoriert.

Bandkompression

Theorem [Bandkompression]
Für jedes $c \in \mathbb{R}^+$ und jede Speicherfunktion $S(n)$ gilt:

$$\textbf{DSPACE}(S(n)) = \textbf{DSPACE}(cS(n))$$
$$\textbf{NSPACE}(S(n)) = \textbf{NSPACE}(cS(n))$$

Beweis Eine Richtung ist trivial. Die andere geht, indem man den Inhalt einer festen Anzahl r ($> \frac{2}{c}$) von benachbarten Zellen auf dem Band als **ein neues Symbol** darstellt. **Die Zustände der neuen Maschine simulieren die alten Kopfbewegungen als Zustandsübergänge** (innerhalb des neuen Symbols). D.h. für r Zellen der alten Maschine werden nun nur maximal 2 benutzt: im ungünstigsten Fall, wenn man von einem Block in den benachbarten geht.

Zeitbeschleunigung

Theorem [Zeitbeschleunigung]

Für jedes $c \in \mathbb{R}^+$ und jede Zeitfunktion $T(n)$ mit $\lim_{n\to\infty} \frac{T(n)}{n} = \infty$ gilt:

$$\mathbf{DTIME}(T(n)) = \mathbf{DTIME}(cT(n))$$

$$\mathbf{NTIME}(T(n)) = \mathbf{NTIME}(cT(n))$$

Beweis Eine Richtung ist trivial. Der Beweis der anderen läuft wieder über die Repräsentation einer festen Anzahl r ($> \frac{4}{c}$) benachbarter Bandzellen durch **neue Symbole**. Im Zustand der neuen Maschine wird wieder kodiert, welches Zeichen und welche Kopfposition (der simulierten, alten Maschine) aktuell ist.

Wenn die alte Maschine simuliert wird, muss die neue nur 4 statt r Schritte machen: 2 um die neuen Felder zu drucken und weitere 2 um den Kopf auf das neue und wieder zurück auf das alte (im schlimmsten Fall) zu bewegen.

Wachstumsrate von DTIME und DSPACE

Wachstumsrate von DTIME, DSPACE

Es sei $T(n)$ eine Zeitfunktion mit $\lim_{n\to\infty} \frac{T(n)}{n} = \infty$, und es sei $S(n)$ eine Speicherfunktion.

(a) Es sei $f(n) = \mathbf{O}(T(n))$. Dann gilt: $\mathbf{DTIME}(f(n)) \subseteq \mathbf{DTIME}(T(n))$.

(b) Es sei $g(n) = \mathbf{O}(S(n))$. Dann gilt: $\mathbf{DSPACE}(g(n)) \subseteq \mathbf{DSPACE}(S(n))$.

Wachstumsrate von DTIME und DSPACE

Definition [P, NP, PSPACE]

$$\begin{aligned} \mathbf{P} &:= \bigcup_{i \geqslant 1} \mathbf{DTIME}(n^i) \\ \mathbf{NP} &:= \bigcup_{i \geqslant 1} \mathbf{NTIME}(n^i) \\ \mathbf{PSPACE} &:= \bigcup_{i \geqslant 1} \mathbf{DSPACE}(n^i) \end{aligned}$$

Wachstumsrate von DTIME und DSPACE

Definition [P, NP, PSPACE]

$$\begin{aligned} \mathbf{P} &:= \bigcup_{i \geqslant 1} \mathbf{DTIME}(n^i) \\ \mathbf{NP} &:= \bigcup_{i \geqslant 1} \mathbf{NTIME}(n^i) \\ \mathbf{PSPACE} &:= \bigcup_{i \geqslant 1} \mathbf{DSPACE}(n^i) \end{aligned}$$

Intuitiv

- Probleme in **P** sind effizient lösbar, jene aus **NP** können in exponentieller Zeit gelöst werden.
- **PSPACE** ist eine sehr große Klasse, weit größer als **P** oder **NP** .

Komplexitätsklassen für Funktionen

Komplexitätsklassen für Funktionen

Eine Funktion $f : \mathbb{N} \to \mathbb{N}$ ist in **P**, falls es
eine DTM $\mathcal{M}$ und ein Polynom $p(n)$ gibt, so dass
für jedes n der Funktionswert $f(n)$ in höchstens
$p(\text{länge}(n))$ Schritten von $\mathcal{M}$ berechnet wird.

Dabei gilt $\text{länge}(n) = \lg n$, denn man braucht $\lg n$ Zeichen,
um die Zahl n binär darzustellen.

Analog funktioniert dies für alle anderen Komplexitätsklassen.

Beziehungen zwischen den Komplexitätsklassen

Frage:

Was sind die genauen Beziehungen zwischen den Komplexitätsklassen P, NP, PSPACE?

Beziehungen zwischen den Komplexitätsklassen

Frage:

Was sind die genauen Beziehungen zwischen den Komplexitätsklassen P, NP, PSPACE?

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$$

Beziehungen zwischen den Komplexitätsklassen

Frage:

Wie zeigen wir, dass ein gegebenes Problem in einer bestimmten Klasse ist?

Antwort

Reduktion auf ein bekanntes!

Wir brauchen eines, mit dem wir anfangen können: SAT

Komplexitätsklassen

Frage:

Können wir in **NP** Probleme finden, die **die schwierigsten in NP** sind?

Komplexitätsklassen

Frage:

Können wir in **NP** Probleme finden, die **die schwierigsten in NP** sind?

Antwort

Es gibt mehrere Wege, ein schwerstes Problem zu definieren. Sie hängen davon ab, welchen **Begriff von Reduzierbarkeit** wir benutzen.

Für einen gegebenen Begriff von Reduzierbarkeit ist die Antwort: **Ja**.

Solche Probleme werden **vollständig in der gegebenen Klasse** bezüglich des Begriffs der Reduzierbarkeit genannt.

Reduktion

Definition (Polynomial-Zeit-Reduzibilität)

Seien L_1, L_2 Sprachen.

L_1 ist Polynomial-Zeit reduzibel auf L_2, bezeichnet mit $L_1 \preceq_{\text{pol}} L_2$,
wenn es eine **Polynomial-Zeit beschränkte DTM** gibt,
die für jede Eingabe w eine Ausgabe $f(w)$ erzeugt, so dass

$$w \in L_1 \text{ gdw } f(w) \in L_2$$

Reduktion

Lemma [Polynomial-Zeit-Reduktionen]

1. Sei L_1 Polynomial-Zeit-reduzibel auf L_2 ($L_1 \preceq_{\text{pol}} L_2$). Dann gilt

   Wenn L_2 in **NP** ist dann ist auch L_1 in **NP**

   Wenn L_2 in **P** ist dann ist auch L_1 in **P**

2. Die Komposition zweier Polynomial-Zeit-Reduktionen ist wieder eine Polynomial-Zeit-Reduktion.

NP

Theorem.
Eine Sprache L is in **NP** genau dann wenn es eine Sprache L' in **P** und ein $k \geq 0$ gibt, so das für alle $w \in \Sigma$ gilt:

$$w \in L \text{ gdw. es gibt ein } c : \langle w, c \rangle \in L' \text{ und } |c| < |w|^k.$$

c wird **Zeuge** (*witness* oder Zertifikat/*certificate*) von w in L genannt.

Eine DTM, die die Sprache L' akzeptiert, wird **Prüfer** (*verifier*) von L genannt.

Wichtig:
Ein Entscheidungsproblem ist in **NP** genau dann wenn **jede Ja-Instanz ein kurzes Zertifikat** hat (d.h. seine Länge polynomial in der Länge der Eingabe ist), welche in polynomial-Zeit verifiziert werden kann.