Grundlagen der Theoretischen Informatik

Komplexitätstheorie (II)

1.07.2021

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

Bis jetzt

- DTIME($f(n)$), NTIME($f(n)$),
- DSPACE($f(n)$), NSPACE($f(n)$),
- Entscheidbarkeit / NTM vs. DTM / Zeit vs. Speicher
- Konstante Faktoren werden ignoriert
- P, NP, PSPACE
- P $\subseteq$ NP $\subseteq$ PSPACE

Bis jetzt

Definition [P, NP, PSPACE]

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

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

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:

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.

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.

Beweis 1. b) Annahme $L_2 \in P$. Dann existiert $k \in \mathbb{N}$ und eine n^k-Zeitbeschränkte DTM $\mathcal{M}_2$, die L_2 akzeptiert.

Da $L_1 \preceq_{\text{pol}} L_2$, gibt es eine **polynomial-Zeit beschränkte DTM** $\mathcal{M}_f$, die für jede Eingabe w eine Ausgabe $f(w)$ erzeugt, so dass $w \in L_1$ gdw $f(w) \in L_2$.

Sei $\mathcal{M}_1 = \mathcal{M}_f\mathcal{M}_2$. $\mathcal{M}_1$ akzeptiert L_1 und ist polynomial-Zeitbeschränkt: $\mathcal{M}_f$ erzeugt $f(w)$ aus w in polynomieller Zeit; $\mathcal{M}_2$ entscheidet ob $f(w) \in L_2$ in polynomieller Zeit.

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.

Beweis 1. a) Annahme $L_2 \in NP$. Dann existiert $k \in \mathbb{N}$ und eine n^k-Zeitbeschränkte NTM $\mathcal{M}_2$, die L_2 akzeptiert.

Da $L_1 \preceq_{\text{pol}} L_2$, gibt es eine **polynomial-Zeit beschränkte DTM** $\mathcal{M}_f$, die für jede Eingabe w eine Ausgabe $f(w)$ erzeugt, so dass $w \in L_1$ gdw $f(w) \in L_2$.

Sei $\mathcal{M}_1 = \mathcal{M}_f\mathcal{M}_2$. $\mathcal{M}_1$ akzeptiert L_1 und ist polynomial-Zeitbeschränkt: $\mathcal{M}_f$ erzeugt $f(w)$ aus w in polynomieller Zeit; $\mathcal{M}_2$ entscheidet ob $f(w) \in L_2$ in polynomieller Zeit.

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.

Beweis 2. Annahme: $L_1 \preceq_{\text{pol}} L_2$ und $L_2 \preceq_{\text{pol}} L_3$

Da $L_1 \preceq_{\text{pol}} L_2$, gibt es eine **polynomial-Zeit beschränkte DTM** $\mathcal{M}_{f_1}$, die für jede Eingabe w eine Ausgabe $f_1(w)$ erzeugt, so dass $w \in L_1$ gdw $f_1(w) \in L_2$.

Da $L_2 \preceq_{\text{pol}} L_3$, gibt es eine **polynomial-Zeit beschränkte DTM** $\mathcal{M}_{f_2}$, die für jede Eingabe w eine Ausgabe $f_2(w)$ erzeugt, so dass $w \in L_2$ gdw $f_2(w) \in L_3$.

Dann gibt es eine **polynomial-Zeit beschränkte DTM** $\mathcal{M}_{f_1}$ $\mathcal{M}_{f_2}$, die für jede Eingabe w eine Ausgabe $f_2(f_1(w))$ erzeugt, so dass:

$$w \in L_1 \text{ gdw } f_1(w) \in L_2 \text{ gdw } f_2(f_1(w)) \in L_3$$

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.

Vollständige und harte Probleme

Vollständige und harte Probleme

Definition [NP-vollständig, NP-hart]

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn
  jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
  1. in **NP** ist ($L \in \mathbf{NP}$), und
  2. **NP**-hart ist

Vollständige und harte Probleme

Definition [NP-vollständig, NP-hart]

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn
  jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
  1. in **NP** ist ($L \in \mathbf{NP}$), und
  2. **NP**-hart ist

Definition [PSPACE-vollständig, PSPACE-hart]

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn
  jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie
  1. in **PSPACE** ist ($L \in \mathbf{PSPACE}$) und
  2. **PSPACE**-hart ist

Vollständige und harte Probleme

Bemerkenswert

- Wenn gezeigt werden kann, dass auch nur ein einziges **NP**-hartes Problem in **P** liegt, dann ist $\mathbf{P} = \mathbf{NP}$.
- Wenn $\mathbf{P} \neq \mathbf{NP}$ gilt, dann ist kein einziges **NP**-vollständiges Problem in polynomieller Zeit lösbar.

Vollständige und harte Probleme

Bemerkenswert

- Wenn gezeigt werden kann, dass auch nur ein einziges **NP**-hartes Problem in **P** liegt, dann ist $\mathbf{P} = \mathbf{NP}$.
- Wenn $\mathbf{P} \neq \mathbf{NP}$ gilt, dann ist kein einziges **NP**-vollständiges Problem in polynomieller Zeit lösbar.

Eine Million Euro für den, der das „P = NP“-Problem löst!
(Millenium Probleme)

Vollständige und harte Problem

Wie zeigt man NP-Vollständigkeit?

Um zu zeigen, dass eine Sprache L **NP**-vollständig ist:

- Zeige, dass $L \in \mathbf{NP}$
- Finde bekanntermaßen **NP**-vollständige Sprache L' und
- reduziere sie auf L:

$$L' \preceq_{\text{pol}} L$$

Das genügt, da jede Sprache aus **NP** auf L' reduzierbar ist und wegen $L' \preceq_{\text{pol}} L$ dann auch auf L.

Hierfür häufig verwendet:
SAT-Problem, d.h.

$$L' = L_{\text{sat}} = SAT = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$$

Cook's Theorem

Theorem $SAT = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$ ist NP-vollständig.

Cook's Theorem

Theorem $SAT = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$ ist NP-vollständig.

Beweis: (Idee)

Zu zeigen: (1) $SAT \in NP$

(2) für alle $L \in NP$, $L \preceq_{\text{pol}} SAT$

Cook's Theorem

Theorem $SAT = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$ ist NP-vollständig.

Beweis: (Idee)

Zu zeigen: (1) $SAT \in NP$

(2) für alle $L \in NP$, $L \preceq_{\text{pol}} SAT$

(1) Gegeben sei F. Man kann in polynomieller Zeit bestimmen, ob F eine aussagenlogische Formel ist. Falls F aussagenlogische Formel: Wertebelegung $\mathcal{A}$ "raten", in polynomieller Zeit zeigen, dass $\mathcal{A}(F) = 1$.

Cook's Theorem

Theorem $SAT = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$ ist NP-vollständig.

Beweis: (Idee)

Zu zeigen: (1) $SAT \in NP$

(2) für alle $L \in NP$, $L \preceq_{\text{pol}} SAT$

(1) Gegeben sei F. Man kann in polynomieller Zeit bestimmen, ob F eine aussagenlogische Formel ist. Falls F aussagenlogische Formel: Wertebelegung $\mathcal{A}$ "raten", in polynomieller Zeit zeigen, dass $\mathcal{A}(F) = 1$.

(2) Sei $L \in NP$. Dann existiert eine polynomiell zeitgebundene NTM $\mathcal{M}$, mit $L(\mathcal{M}) = L$. Für $\mathcal{M}$ und w kann man eine aussagenlogische Sprache definieren und in polynomieller Zeit eine Formel $T_{M,w}$ finden, so dass

$$w \in L(\mathcal{M}) \quad \text{gdw.} \quad T_{M,w} \text{ erfüllbar ist.}$$

Stephen Cook

Stephen Arthur Cook (geboren 1939)

- Einer der bedeutendsten Forschern
  in der Komplexitätstheorie.
- 1971 'The Complexity of Theorem Proving Procedures'
  - formalisiert die Polynomialzeitreduktion
  - begründet mit dem Satz von Cook das
    Problem der NP-Vollständigkeit
    und im Besonderen das P-NP-Problem.
- Professor der Informatik an der
  University of Toronto in Kanada.
- 1982: Turing award

Vollständige und harte Probleme

Nota Bene: Es gibt NP-harte Probleme, die nicht in NP sind
(und z.B. sogar nicht entscheidbar sein können).

Vollständige und harte Probleme

Nota Bene: Es gibt NP-harte Probleme, die nicht in NP sind
(und z.B. sogar nicht entscheidbar sein können).

Beispiel: $SAT \preceq_{\text{pol}} SAT_{\text{PL}}$, wobei:

$$\begin{aligned} SAT &= \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\} \\ SAT_{\text{PL}} &= \{w \mid w \text{ ist eine erfüllbare Formel in der Prädikatenlogik}\} \end{aligned}$$

Sei $f : \Sigma^* \to \Sigma^*$ mit $\begin{cases} f(w) = w & \text{falls } w \text{ eine aussagenlogische Formel ist} \\ f(w) = \varepsilon \text{ sonst} \end{cases}$

(kann von einer polynomial-Zeit beschränkte DTM berechnet werden).

Dann: $F \in SAT$ gdw. F ist eine erfüllbare aussagenlogische Formel
gdw. $f(F) = F$ ist eine erfüllbare Formel in der Prädikatenlogik
gdw. $f(F) \in SAT_{\text{PL}}$

- SAT ist NP-vollständig, also ist SAT_{PL} NP-hart.
- $SAT_{\text{PL}} \notin$ NP. (SAT_{PL} ist nicht entscheidbar: die Menge aller erfüllbaren prädikatenlogischen Formeln ist nicht entscheidbar).

Abgschlossenheit der Komplexitätsklassen

Abgschlossenheit der Komplexitätsklassen

P, PSPACE sind abgeschlossen unter Komplement

Alle Komplexitätsklassen, die mittels **deterministischer Turing-Maschinen** definiert sind, sind **abgeschlossen unter Komplement-Bildung**

Denn:
Wenn eine Sprache L dazu gehört, dann auch ihr Komplement
(einfach die alte Maschine ausführen und die Ausgabe invertieren).

Abgschlossenheit der Komplexitätsklassen

Abgeschlossenheit von NP unter Komplement

Frage:

Ist **NP** abgeschlossen unter Komplementbildung?

Abgschlossenheit der Komplexitätsklassen

Abgeschlossenheit von NP unter Komplement

Frage:

Ist **NP** abgeschlossen unter Komplementbildung?

Antwort:

Keiner weiß es!

Die Kompexitätsklasse co-NP

Definition [co-NP]

co-NP ist die Klasse der Sprachen deren Komplemente in **NP** liegen:

$$\textbf{co-NP} = \{L \mid \overline{L} \in \textbf{NP}\}$$

Beziehungen zwischen Komplexitätsklassen

Die folgenden Beziehungen sind momentan noch unbekannt

1. **P** $=?$ **NP**.
2. **NP** $=?$ **co-NP**.
3. **P** $=?$ **PSPACE**.
4. **NP** $=?$ **PSPACE**.