3 Probabilistische Komplexitätsklassen

3.1 Probabilistische Turingmaschinen

3.1 Wir gehen davon aus, dass die Konzepte deterministischer und nichtdeterministischer Turingmaschinen im wesentlichen bekannt sind.

Die Menge der („echten", s.u.) Zustände wird mit S bezeichnet und das Bandalphabet mit B. Das Blanksymbol wird gegebenenfalls als $\square$ geschrieben. $A \subseteq B - \{\square\}$ ist das Eingabealphabet.

Wir beschränken uns auf TM mit einem Band und einem Kopf. In diesem Fall ist bei einer deterministischen TM (DTM) die Überführungsfunktion von der Form $\delta : S \times B \to (S \cup \{\text{YES}, \text{NO}\}) \times B \times \{-1, 0, 1\}$. Für die Endzustände YES und NO müssen keine Regeln angegeben werden.

Eingaben werden auf dem ansonsten mit $\square$-Symbolen beschrifteten Band zur Verfügung gestellt, wobei sich der Kopf anfangs auf dem ersten Eingabesymbol befindet. Wir beschränken uns (weitgehend) auf Entscheidungsprobleme (i. e. die Erkennung formaler Sprachen). Dabei wird das Ergebnis (Annahme oder Ablehnung) am erreichten Endzustand (YES oder NO) abgelesen.

Für eine nichtdeterministische TM (NTM) ist die Überführungsfunktion von der Form $\delta : S \times B \to 2^{(S \cup \{\text{YES},\text{NO}\}) \times B \times \{-1,0,1\}}$.

Die von einer TM T *akzeptierte Sprache* ist

$$L(T) = \{w \in A^+ \mid \text{ es gibt eine akzeptierende Berechnung von } T \text{ für Eingabe } w\} .$$

Wie schon zu Beginn des ersten Kapitels erwähnt, kann man für probabilistische Turingmaschinen formal unterschiedliche Definitionen geben, die aber inhaltlich äquivalent sind.

3.2 DEFINITION Eine *probabilistische Turingmaschine* (PTM) ist formal wie eine nichtdeterministische TM festgelegt, jedoch mit der zusätzlichen Einschränkung, dass für alle Paare (s, b) genau eine oder zwei mögliche Alternativen für den nächsten Schritt vorliegen: $1 \leqslant |\delta(s, b)| \leqslant 2$. Und es wird vereinbart, dass eine PTM in einer Berechnungssituation, in der zwei Möglichkeiten vorliegen, jede mit gleicher Wahrscheinlichkeit $1/2$ wählt. ◇

3.3 Diese „lokale" Quantifizierung zusammen mit der anderen Sichtweise, sich z. B. für die Wahrscheinlichkeit des Akzeptierens einer Eingabe (also quantitative Eigenschaften des globalen Verhaltens) zu interessieren, macht den wesentlichen Unterschied zu nichtdeterministischen TM aus.

Man beachte auch, dass $1/2$ ein „harmloser" Wert ist. Es ist nicht erlaubt, andere Wahrscheinlichkeiten p zu wählen, bei denen etwa in die Dezimalbruchentwicklung von p die Lösung des Halteproblems hineinkodiert ist.

3.4 DEFINITION Wir wollen sagen, dass eine PTM in *Normalform* vorliege, wenn sie eine PTM ist, die die folgenden Bedingungen erfüllt:

- Für jedes Paar $(s, b) \in S \times B$ gibt es *genau* zwei mögliche Alternativen: $|\delta(s, b)| = 2$.

- Für jede Eingabe sind alle Berechnungen gleich lang. ◇

3.5 Für PTM in Normalform bereitet es dann auch keine Schwierigkeiten die *Zeitkomplexität* $t(n)$ auf die naheliegende Weise zu definieren. Eine TM arbeite in *Polynomialzeit*, wenn es ein Polynom $p(n)$ gibt, so dass $t(n) \leqslant p(n)$ ist.

Der Berechnungsbaum einer PTM in Normalform ist also für jede Eingabe eine voller balancierter binärer Baum.

3.2 Komplexitätsklassen

3.6 Wir benutzen die folgenden üblichen Abkürzungen für wichtige Komplexitätsklassen:

Klasse $\mathcal{C}$	Kriterium für $L \in \mathcal{C}$
P	L kann von einer DTM in Polynomialzeit erkannt werden
NP	L kann von einer NTM in Polynomialzeit erkannt werden
PSPACE	L kann von einer DTM oder NTM in polynomialem Platz erkannt werden

Für eine Komplexitätsklasse $\mathcal{C}$ sei co$-\mathcal{C} = \{L \mid \bar{L} \in \mathcal{C}\}$. Bekanntlich ist $\mathbf{P} =$ co$-\mathbf{P}$ und $\mathbf{P} \subseteq \mathbf{NP} \cap$ co$-\mathbf{NP}$. Die Beziehung zwischen **NP** und co$-$**NP** ist ungeklärt. Man weiß, dass $\mathbf{NP} \cup$ co$-\mathbf{NP} \subseteq$ **PSPACE** ist.

Eine *Polynomialzeitreduktion* einer Sprache L_1 auf eine Sprache L_2 ist eine Abbildung $f : A^+ \to A^+$, die in Polynomialzeit berechnet werden kann und für die für alle $w \in A^+$ gilt:

$$w \in L_1 \iff f(w) \in L_2 \,.$$

In einer solchen Situation folgt z. B. aus $L_2 \in \mathbf{NP}$ sofort auch $L_1 \in \mathbf{NP}$.

Eine Sprache H ist **NP**-*hart*, wenn jede Sprache aus **NP** auf H polynomialzeitreduziert werden kann. Eine Sprache ist **NP**-*vollständig*, wenn sie **NP**-hart und aus **NP** ist.

Im Hinblick auf die nachfolgenden Definitionen randomisierter Komplexitätsklassen schreiben wir noch einmal die für **P** etwas anders auf. Es ist genau dann $L \in \mathbf{P}$, wenn es eine DTM T gibt, die in Polynomialzeit arbeitet und für die für alle Eingaben $w \in A^+$ gilt:

- $w \in L \implies T$ akzeptiert w
- $w \notin L \implies T$ akzeptiert w nicht

Wir definieren nun einige wichtige randomisierte Komplexitätsklassen.

3.7 Definition Im folgenden sind alle Implikationen als für alle $w \in A^+$ quantifiziert zu verstehen.

1. $L \in \mathbf{RP}$, wenn es eine Polynomialzeit-PTM T gibt, für die gilt:
   - $w \in L \implies \mathbf{P}(T \text{ akzeptiert } w) \geqslant 1/2$
   - $w \notin L \implies \mathbf{P}(T \text{ akzeptiert } w) = 0$
2. $\mathbf{ZPP} = \mathbf{RP} \cap$ co$-\mathbf{RP}$.
3. $L \in \mathbf{BPP}$, wenn es eine Polynomialzeit-PTM T gibt, für die gilt:

- $w \in L \implies \mathbf{P}(T \text{ akzeptiert } w) > 3/4$
- $w \notin L \implies \mathbf{P}(T \text{ akzeptiert } w) < 1/4$

4. $L \in \mathbf{PP}$, wenn es eine Polynomialzeit-PTM T gibt, für die gilt:
   - $w \in L \implies \mathbf{P}(T \text{ akzeptiert } w) > 1/2$
   - $w \notin L \implies \mathbf{P}(T \text{ akzeptiert } w) \leqslant 1/2$ ◇

Im Falle von **BPP** ist also die Fehlerwahrscheinlichkeit in beiden Fällen ($w \in L$ bzw. $w \notin L$) kleiner als $1/4$.

Man kann zeigen, dass man bei der Definition von **PP** bei $w \notin L$ statt $\leqslant 1/2$ auch $< 1/2$ hätte schreiben können, ohne inhaltlich etwas zu ändern. Insofern darf man sagen, dass im Falle von **PP** die Fehlerwahrscheinlichkeit kleiner $1/2$ ist.

3.8 Bei **RP** besteht die Möglichkeit *einseitiger* Fehler, bei **BPP** und **PP** die *zweiseitiger* Fehler. Wir werden sehen, dass man im Falle von **ZPP** Maschinen konstruieren kann, deren Antworten nie fehlerhaft sind.

Generell bezeichnet man randomisierte Algorithmen, die eine endliche erwartete Laufzeit haben und deren Antworten nie falsch sind, als *Las Vegas*-Algorithmen. Algorithmen, die möglicherweise fehlerhafte Antworten liefern, heißen auch *Monte Carlo*-Algorithmen.

3.9 Beispiel. Der Primzahltest von Rabin (1976) zeigt, dass die Sprache der Primzahlen in co−**RP** liegt. Viel schwieriger war es, nachzuweisen, dass sie auch in **RP** (und damit in **ZPP**) ist (Adleman und Huang 1987). Inzwischen weiß man, dass PRIMES sogar in **P** liegt (Agrawal, Kayal und Saxena 2002).

3.10 Für PTM, die in Normalform sind, kann man die Klassen **RP**, **BPP** und **PP** auch alternativ wie folgt festlegen:

1. $L \in \mathbf{RP}$, wenn es eine Polynomialzeit-PTM T gibt, für die gilt:
   - $w \in L \implies$ mindestens $1/2$ aller möglichen Berechnungen liefern Antwort YES.
   - $w \notin L \implies$ alle Berechnungen liefern Antwort NO.
2. $L \in \mathbf{BPP}$, wenn es eine Polynomialzeit-PTM T gibt, für die gilt:
   - $w \in L \implies$ mindestens $3/4$ aller möglichen Berechnungen liefern Antwort YES.
   - $w \notin L \implies$ mindestens $3/4$ aller möglichen Berechnungen liefern Antwort NO.
3. $L \in \mathbf{PP}$, wenn es eine Polynomialzeit-PTM T gibt, für die gilt:
   - $w \in L \implies$ mehr als die Hälfte aller möglichen Berechnungen liefern Antwort YES.
   - $w \notin L \implies$ mindestens die Hälfte aller möglichen Berechnungen liefern Antwort NO.

Eine PTM, die die Zugehörigkeit der von ihr erkannten Sprache zu **RP** (resp. **BPP** oder **PP**) belegt, wollen wir auch eine **RP**-PTM (resp. **BPP**-PTM oder **PP**-PTM) nennen.

Im Fall von **PP** wird die Entscheidung sozusagen durch absolute Mehrheit getroffen.

3.11 Die Definition von **RP** beinhaltet ein Problem. Es ist für eine vorgegebene PTM (gleichgültig, ob in Normalform oder nicht) nicht ersichtlich, ob sie ein Beweis dafür ist, dass die von ihr erkannte Sprache in **RP** ist. Das ist sogar *unentscheidbar*.

Die analoge Aussage gilt auch für **BPP**.

3.12 Satz. *Für jedes Polynom $q(n) \geqslant 1$ kann man in der Definition von* **RP** *anstelle von $1/2$ auch $1-2^{-q(n)}$ einsetzen, also Fehlerwahrscheinlichkeit $2^{-q(n)}$ statt $1/2$ fordern, ohne an der Klasse etwas zu verändern.*

3.13 Beweis. Es sei R eine **RP**-PTM, die L mit Fehlerwahrscheinlichkeit $1/2$ erkennt. Eine **RP**-PTM R', die L mit Fehlerwahrscheinlichkeit $2^{-q(n)}$ erkennt, kann wie folgt konstruiert werden.

Es sei w eine Eingabe der Länge n. R' verwaltet einen Zähler, der mit $q(n)$ initialisiert wird. In einer Schleife wird er auf Null heruntergezählt und dabei jedes Mal eine Berechnung von R für die Eingabe w simuliert. Wenn R bei mindestens einem Versuch YES liefern würde, beendet R' seine Berechnung mit Antwort YES. Andernfalls antwortet R' mit NO.

Ist nun $w \in L(R)$, dann ist die Wahrscheinlichkeit, dass R' die falsche Antwort liefert, $(1/2)^{q(n)} = 2^{-q(n)}$. Ist $w \notin L(R)$, dann antwortet R und daher auch R' immer mit NO.

Bezeichnet $p(n)$ die Laufzeit von R, so ist die von R' gerade $p(n)q(n)$, also ebenfalls polynomiell. ∎

3.14 Satz. *Für jedes Polynom $q(n) \geqslant 2$ kann man in der Definition von* **BPP** *anstelle der Fehlerwahrscheinlichkeit von $1/4$ auch $2^{-q(n)}$ einsetzen, ohne an der Klasse etwas zu verändern.*

3.15 Beweis. Es sei R eine **BPP**-PTM, die L mit Fehlerwahrscheinlichkeit $1/4$ erkennt. Eine **BPP**-PTM R', die L mit Fehlerwahrscheinlichkeit $2^{-q(n)}$ erkennt, kann wie folgt konstruiert werden.

Es sei w eine Eingabe der Länge n. Es sei $m = 2q+1$ eine ungerade Zahl, die weiter unten geeignet festlegt wird.

R' verwaltet einen Zähler, der mit m (einer ungeraden Zahl) initialisiert wird. In einer Schleife wird er auf Null heruntergezählt und dabei jedes Mal eine Berechnung von R für die Eingabe w simuliert. Es wird gezählt, wie oft R Antwort YES liefern würde, und wie oft Antwort NO.

Die Antwort von R' ist die, die von R häufiger gegeben wurde.

Wir rechnen nun nach, dass R' die gewünschten Eigenschaften hat, wenn q geeignet gewählt wird.

Die Wahrscheinlichkeit, eine falsche Antwort zu erhalten, ist höchstens

$$\begin{aligned}
\sum_{i=0}^{q} \binom{m}{i} \left(\frac{3}{4}\right)^i \left(\frac{1}{4}\right)^{m-i} &\leqslant \sum_{i=0}^{q} \binom{m}{i} \left(\frac{3}{4}\right)^i \left(\frac{1}{4}\right)^{m-i} \left(\frac{3/4}{1/4}\right)^{m/2-i} \\
&= \sum_{i=0}^{q} \binom{m}{i} \left(\frac{3}{4}\right)^{m/2} \left(\frac{1}{4}\right)^{m/2} \\
&= \left(\frac{3}{16}\right)^{m/2} \sum_{i=0}^{q} \binom{m}{i} \\
&\leqslant \left(\frac{3}{16}\right)^{m/2} 2^{m-1} = \frac{1}{2}\left(\frac{3}{4}\right)^{m/2}
\end{aligned}$$

Die Wahrscheinlichkeit für eine richtige Antwort ist also größer gleich $1 - \frac{1}{2}\left(\frac{3}{4}\right)^{m/2}$. Man wählt nun m so, dass gilt (log sei zur Basis 2 genommen):

$$
\begin{aligned}
& 1-\frac{1}{2}\left(\frac{3}{4}\right)^{m/2} \geqslant 1-2^{-q(n)} \\
\iff\quad & \frac{1}{2}\left(\frac{3}{4}\right)^{m/2} \leqslant 2^{-q(n)} \\
\iff\quad & 2\left(\frac{4}{3}\right)^{m/2} \geqslant 2^{q(n)} \\
\iff\quad & \left(\frac{4}{3}\right)^{m/2} \geqslant 2^{q(n)-1} \\
\iff\quad & \frac{m}{2}(2-\log 3) \geqslant q(n)-1 \\
\iff\quad & m \geqslant \frac{2(q(n)-1)}{2-\log 3}
\end{aligned}
$$

Wie man sieht, kann eine geeignete Anzahl von Wiederholungen linear in $q(n)$ gewählt werden, so dass mit der Laufzeit von R auch die von R′ polynomiell ist. ∎

3.16 Satz. *Wenn* $L \in$ **ZPP** *ist, dann gibt es eine PTM, für die der Erwartungswert der Laufzeit polynomiell ist und die* L *entscheidet (d. h.* immer *richtige Antworten liefert).*

3.17 Beweis. Es sei R eine **RP**-PTM und $\bar{R}$ eine **RP**-PTM für $\bar{L}$, die bei jedem Lauf für eine Eingabe w als Antwort liefern. Der folgende Algorithmus leistet das Gewünschte:

⟨*Eingabe: Wort* w⟩
⟨*Ausgabe:* yes *falls* $w \in L$, no *falls* $w \notin L$⟩
repeat
  $r \leftarrow R(w)$
  $r' \leftarrow$ **not** $\bar{R}(w)$ ⟨$\bar{R} = \text{yes} \Longrightarrow w \in \bar{L}$, *also* $w \notin L$⟩
until $r = r'$
return r

Dass die Antwort dieses Algorithmus *immer* korrekt ist, ergibt sich aus der Tatsache, dass im Fall $w \in L$ (bzw. $w \notin L$, i. e. $w \in \bar{L}$) die Behauptung von $\bar{R}$ (bzw. R) garantiert richtig ist.

Es sei $p(n)$ das Maximum der (polynomiellen) Laufzeiten von R bzw. $\bar{R}$. Der Erwartungswert für die Laufzeit des obigen Algorithmus ist nach oben beschränkt durch

$$\frac{1}{2}2p(n)+\frac{1}{4}4p(n)+\frac{1}{8}6p(n)+\cdots = 2p(n)\sum_{i=1}^{\infty} 2^{-i}i = 4p(n)\,.$$

(Hinweis: $2-\sum_{i=0}^{k} i2^{-i} = (k+2)2^{-k}$.) ∎

3.18 Bei obigem Algorithmus kann es für eine Eingabe passieren, dass *nie* ein Ergebnis geliefert wird.

Deshalb treffen manche Autoren (z. B. Goos 1999) bei Las Vegas-Algorithmen noch die Unterscheidung, ob sie immer terminieren oder nicht. Im ersteren Fall sprechen sie dann genauer von *Macao*-Algorithmen.

3.3 Beziehungen zwischen Komplexitätsklassen

3.19 Gegenstand dieses Abschnitts sind die in Abbildung 3.1 dargestellten Inklusionsbeziehungen. Ein Pfeil von **A** nach **B** bedeutet, dass $\mathbf{A} \subseteq \mathbf{B}$ ist. Inklusionsbeziehungen, die sich durch Transitivität ergeben, sind nicht dargestellt. Ansonsten bedeutet die Abwesenheit eines Pfeiles in diesem Diagramm, dass man nicht weiß, ob eine Inklusionsbeziehung besteht oder nicht.

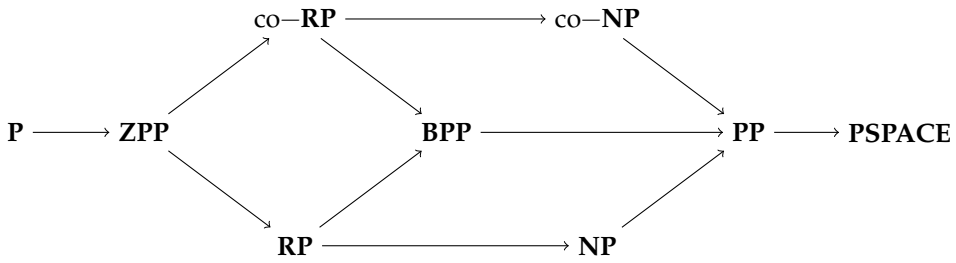

Abbildung 3.1: Beziehungen zwischen Komplexitätsklassen.

Da noch nicht einmal klar ist, ob die Inklusion $\mathbf{P} \subseteq \mathbf{PSPACE}$ echt ist oder nicht, weiß man das natürlich auch von keiner der „dazwischen liegenden“ Inklusionen.

Da jede deterministische TM gleichzeitig auch eine **RP**-PTM und eine co−**RP**-PTM ist, und aufgrund der Definition von **ZPP** ist zunächst einmal klar:

3.20 Satz. $\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{RP}$ *und* $\mathbf{ZPP} \subseteq$ co−**RP**.

Denkt man an die alternativen Beschreibungen der Komplexitätsklassen mit Hilfe von Normalform-PTM in Punkt 3.10, dann ist klar, dass gilt:

3.21 Satz. $\mathbf{RP} \subseteq \mathbf{NP}$ *und* co−$\mathbf{RP} \subseteq$ co−**NP**.

Nach Satz 3.12 darf man bei der Definition von **RP** (bzw. co−**RP**) ohne Beschränkung der Allgemeinheit eine Fehlerwahrscheinlichkeit von $1/4$ einsetzen (man wähle $q(n) = 2$). Also gilt:

3.22 Satz. $\mathbf{RP} \subseteq \mathbf{BPP}$ *und* co−$\mathbf{RP} \subseteq \mathbf{BPP}$.

Offensichtlich gilt:

3.23 Satz. $\mathbf{BPP} \subseteq \mathbf{PP}$.

Für den Nachweis der beiden letzten Inklusionen muss man wieder etwas arbeiten.

3.24 Satz. $\mathbf{NP} \subseteq \mathbf{PP}$ *und* co−$\mathbf{NP} \subseteq \mathbf{PP}$.

3.25 Beweis. Wir beschränken uns auf den Nachweis der ersten Inklusion.

Es sei N eine Polynomialzeit-NTM in Normalform. Daraus wird eine PTM N' konstruiert, indem am Ende jeder Berechnung ein weiterer Schritt mit zwei Alternativen angehängt wird. Im einen Fall wird die ursprüngliche Antwort von N geliefert, im anderen Fall auf jeden Fall yes.

Ist eine Eingabe $w \in L(N)$, dann gibt es bei N mindestens eine akzeptierende Berechnung. Folglich ist bei N' mehr als die Hälfte aller Berechnungen akzeptierend. Ist dagegen $w \notin L(N)$, dann ist nur genau die Hälfte aller Berechnungen akzeptierend.

Da mit N offensichtlich auch N' in Polynomialzeit arbeitet, ist N' die gesuchte Maschine. ■

3.26 Satz. **PP** $\subseteq$ **PSPACE**.

3.27 Beweis. Es sei P eine **PP**-PTM in Normalform mit Zeitkomplexität $p(n)$ und w eine beliebige Eingabe. Der folgende naheliegende deterministische Algorithmus leistet das Gewünschte:

⟨*Eingabe:* w⟩
$a \leftarrow 0$ ⟨*für Zählung der akzeptierenden Berechnungen von* P⟩
$k \leftarrow p(|w|)$ ⟨*Anzahl Schritte von* P⟩
⟨*für alle Bitfolgen der Länge* k⟩
for $(b_k b_{k-1} \cdots b_1) \leftarrow (000\cdots0)$ **to** $(111\cdots1)$ **do**
  $r \leftarrow$ Simulation von $P(w)$ mit Entscheidungen gemäß den b_i
  if $r =$ YES **then** $a \leftarrow a + 1$ **fi**
od
if $a > 2^{k-1}$ **then**
  return YES
else
  return NO
fi

Der Platzbedarf dieses Algorithmus wird dominiert von dem für die Bits b_i und dem für die Simulationen von $P(w)$. Die Anzahl k der Bits ist polynomiell in der Länge der Eingabe. Jede der Simulationen dauert polynomiell lange. In dieser Zeit kann daher auch nicht auf mehr als polynomiell viele Speicherplätze zugegriffen werden. Also ist der Gesamtplatzbedarf der obigen Prozedur polynomiell. ■

Zusammenfassung

1. Es gibt randomisierte Algorithmen ohne Fehler, mit einseitigem und mit zweiseitigem Fehler.
2. Die Fehlerwahrscheinlichkeit kann man relativ einfach reduzieren, benötigt dafür aber „deutlich“ mehr Zufallsbits.

Literatur

Adleman, L. M. und A. M.-D. Huang (1987). „Recognizing Primes in Random Polynomial Time“. In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, S. 462–469 (siehe S. 22).

Agrawal, Manindra, Neeraj Kayal und Nitin Saxena (Aug. 2002). *PRIMES is in P*. Report. Kanpur-208016, India: Department of Computer Science und Engineering, Indian Institute of Technology Kanpur. URL: http://www.cse.iitk.ac.in/news/primality.pdf (siehe S. 22).

Goos, Gerhard (1999). *Vorlesung über Informatik*. Bd. 2. Heidelberg: Springer (siehe S. 24).
Rabin, Michael O. (1976). „Probabilistic Algorithms“. In: *Algorithms and Complexity*. Hrsg. von J. F. Traub. Academic Press, S. 21–39 (siehe S. 22).

