# Komplexitätsklassen

Eine Komplexitätsklasse ist gegeben durch

- eine Computermodell (PCP, RAM, multi-$ TM, ... ),
- einen Berechnungsmodus (probabilistisch, deterministisch, nichtdeterministisch, ... ),
- Ressourcen (Zeit, Platz, ... ), und
- eine Begrenzungsfunktion $f$.

Eine Komplexitätsklasse ist die Menge aller Sprachen, die durch eine multi-$ TM $M$ entschieden werden, wobei $M$ mit dem entsprechenden Berechnungsmodus arbeitet und zwar so, dass für eine beliebige Eingabe $x$, $M$ immer höchstens $f(|x|)$ Einheiten der entsprechenden Ressource verbraucht.

**Definition:** Sei $f : \mathbb{N}^+ \mapsto \mathbb{N}^+$. Wir nennen $f$ eine ordentliche Komplexitätsfunktion wenn $f$ nicht-absteigend ist und folgendes gilt: Es gibt eine $k$-\$ TM $M_f$ mit Input und Output, sodass bei beliebiger Eingabe $x$, $M_f(x) = \sqcap^{f(|x|)}$ und $M_f$ hält nach $O(|x| + f(|x|))$ Schritten und verbraucht abgesehen vom Input $O(f(|x|))$ viel Platz.

- Beispiele für ordentliche Funktionen: $f(n) = c$, $f(n) = n$, $f(n) = \lceil \log n \rceil$, $(\log n)^2$, $n \log n$, $n^2$, $n^3 + 3n$, $2^n$, $\sqrt{n}$, $n!$

- Wenn $f$ und $g$ ordentlich sind, dann sind es auch $f + g$, $f \cdot g$, und $2^g$ .

**Definition:** Eine TM $M$ sei präzise wenn es Funktionen $f$ und $g$ gibt, sodass für jede Eingabe $x$ gilt, dass $M$ genau nach $f(|x|)$ Schritten hält und jeder \$ (außer eventuellen Input/Output \$s) sei genau $g(|x|)$ Plätze lang.

**Theorem:** Sei $M$ eine TM (deterministisch oder nicht-deterministisch), die eine Sprache $L$ in Zeit (oder Platz) $f(n)$ entscheide, wobei $f(n)$ eine ordentliche Funktion sei. Dann gibt es eine präzise TM $M'$, die $L$ in $O(f(n))$ Schritten/Platzeinheiten entscheidet.

**Beweis:** $M'$ besteht aus einem Teil, der zuerst ein Maßband (yardstick) oder einen Wecker (alarm clock) der Länge $\sqcap^{f(|x|)}$ berechnet und dann $M$ genau $f(|x|)$ Schritte/Platzeinheiten weit simuliert. $\surd$

- Komplexitätsklassen **TIME**$(f)$, **NTIME**$(f)$, **SPACE**$(f)$, und **NSPACE**$(f)$.
- $f$ kann eine Funktionsfamilie mit Parameter $k$ sein; die Komplexitätsklasse ist die Vereinigung über alle $k$.

Einige wichtige Komplexitätsklassen:

$$
\begin{aligned}
\textbf{P} &= \textbf{TIME}(n^k) = \textstyle\bigcup_{j>0} \textbf{TIME}(n^j) \\
\textbf{NP} &= \textbf{NTIME}(n^k) \\
\textbf{PSPACE} &= \textbf{SPACE}(n^k) \\
\textbf{NPSPACE} &= \textbf{NSPACE}(n^k) \\
\textbf{EXP} &= \textbf{TIME}(2^{n^k}) \\
\textbf{L} &= \textbf{SPACE}(\log(n)) \\
\textbf{NL} &= \textbf{NSPACE}(\log(n))
\end{aligned}
$$

## Komplement von nichtdeterministischen Klassen

- Komplement einer Sprache $L \subseteq \Sigma^*$: $\overline{L} = \Sigma^* - L$ .
- Komplement eines Entscheidungsproblems $A$, genannt $A$ COMPLEMENT: Antworte „ja" gdw die Antwort für $A$ „nein" war.
  Beispiel: SAT COMPLEMENT: Gegeben eine Boolsche Formel, ist sie unerfüllbar? (UNSAT, TAUT).
- Für beliebige Komplexitätsklasse $C$ benennen wir mit **co**$C$ die Klasse $\{\overline{L} \,/\, L \in C\}$.
- Alle deterministischen Klassen sind invariant bzgl Komplement, zB **P** $=$ **coP**.
- Gilt auch für **NSPACE** Klassen. $\longrightarrow$ Zu zeigen!
- Für **NTIME** Klassen ist es ein offenes Problem.

## Die Hierarchie Theoreme

- Ein quantitatives Resultat: mit einem ausreichenden Mehr an Zeit kann eine TM beweisbar komplexere Probleme lösen.
- Sei $f(n) \geq n$ eine ordentliche Komplexitätsfunktion. Sei $H_f \stackrel{\text{def}}{=} \{M;x \mid M \text{ akzeptiert } x \text{ nach höchstens } f(|x|) \text{ Schritten}\}$ .

**Lemma:** $H_f \in \mathbf{TIME}(f(n)^3)$ .

**Beweis:** Wir beschreiben eine 4-\$ TM $U_f$, die $H_f$ in $f(n)^3$ Schritten akzeptiert, basierend auf (i) einer universellen TM, (ii) der 1-\$ Simulation einer multi-\$ TM, (iii) der linearen speed-up Maschine, und (iv) $M_f$, die ein Maßband der Länge $f(n)$ berechnet. ▷

Zuerst berechnet $M_f$ auf \$4 die „Wecker-Funktion“ $\sqcap^{f(|x|)}$.

Die $k$\$ TM $M$ kommt, für die universale TM $U_f$ kodiert, auf \$3. Auf \$2 wird der Initialzustand $s$ gespeichert und auf \$1 die Eingabe $\triangleright x$.

$U_f$ simuliert $M$ und geht dabei pro Schritt von $M$ das Wecker Maßband ab. Wenn $M$ dabei die Eingabe akzeptiert, bevor das Wecker Maßband an seinem Ende angelangt ist, dann akzeptiert auch $U_f$, sonst verwirft sie.

Die $k$\$ TM $M$ wird auf einer 1\$ TM simuliert, daher braucht jeder Schritt $O(f(n)^2)$ Zeit.

Insgesamt braucht man $O(f(n)^3)$ Zeit für $f(|x|)$ Schritte.

Mittels des linearen speed-up Tricks (mehrere Symbole auf einmal verarbeiten) kann man die Laufzeit auf höchstens $f(n)^3$ beschränken. $\surd$

**Lemma:** $H_f \notin \mathbf{TIME}(f(\lfloor \frac{n}{2} \rfloor))$ .

**Beweis:** Angenommen, es gäbe eine TM $M_{H_f}$, die $H_f$ in höchstens $f(\lfloor \frac{n}{2} \rfloor)$ Schritten entscheiden kann. Dann sei

$D_f(M)$ : **if** $M_{H_f}(M;M) =$ „ja“ **then** „nein“ **else** „ja“.

$$\left.\begin{array}{l}\text{$D_f$ braucht bei Eingabe $M$ genauso viele Schritte wie $M_{H_f}$}\\ \text{auf $M;M$ mit $|M;M| = 2|M|+1$, dh $f(\lfloor \frac{2|M|+1}{2} \rfloor) = f(|M|)$.}\end{array}\right\}(*)$$

Frage: Akzeptiert $D_f$ dann ihre eigene Beschreibung?

Angenommen $D_f(D_f) =$ „ja“, dh $M_{H_f}(D_f;D_f) =$ „nein“ $\Leftrightarrow$ $D_f;D_f \notin H_f$ . Das bedeutet aber, dass $D_f$ ihren eigenen Code $D_f$ nicht in $f(|D_f|)$ Schritten akzeptiert. Da $D_f$ aber (siehe $(*)$) sicher in $f(|D_f|)$ Schritten hält und nur entweder verwerfen oder akzeptieren kann, muss sie die Eingabe $D_f$ verwerfen, dh $D_f(D_f) =$ „nein“, im Widerspruch zur Annahme dass $D_f(D_f) =$ „ja“.

$\triangleright$

Umgekehrt angenommen $D_f(D_f) =$ „nein“, dh $M_{H_f}(D_f;D_f) =$ „ja“ $\Leftrightarrow D_f;D_f \in H_f$ . Das bedeutet aber, dass $D_f$ ihren eigenen Code $D_f$ in $f(|D_f|)$ Schritten akzeptiert, dh $D_f(D_f) =$ „ja“, dh erneuter Widerspruch.

Es kann daher keine solche TM $M_{H_f}$ mit der angenommenen Eigenschaft geben, da dies auf jeden Fall zu einem Widerspruch führt (Diagonalisierungsargument). $\surd$

**Theorem:** (Das Zeit-Hierarchie Theorem)
Wenn $f(n) \geq n$ eine ordentliche Komplexitätsfunktion ist, dann gilt $\textbf{TIME}(f(n)) \subset \textbf{TIME}((f(2n+1))^3)$ .

**Beweis:**
Wegen dem 1. Lemma: $H_{f(2n+1)} \in \textbf{TIME}((f(2n+1))^3)$ .
Wegen dem 2. Lemma:

$$H_{f(2n+1)} \notin \textbf{TIME}((f(\lfloor \tfrac{2n+1}{2} \rfloor)) = \textbf{TIME}(f(n)) . \surd$$

**Korollar:** **P** $\subset$ **EXP** .

**Beweis:** $n^k = O(2^n)$, daher **P** $\subseteq$ **TIME**$(2^n) \subseteq$ **EXP** . Wegen dem Zeit-Hierarchie Theorem gilt außerdem:

$$\textbf{TIME}(2^n) \subset \textbf{TIME}((2^{2n+1})^3) \subseteq \textbf{TIME}(2^{n^2}) \subseteq \textbf{EXP}\ .\ \surd$$

**Theorem:** (Das Platz-Hierarchie Theorem)
Wenn $f(n)$ eine ordentliche Komplexitätsfunktion ist, dann gilt $\textbf{SPACE}(f(n)) \subset \textbf{SPACE}(f(n)\log f(n))$ .

**Theorem:** (Das Gap Theorem)
Es gibt eine rekursive Funktion $f : \mathbb{N}^+ \mapsto \mathbb{N}^+$, sodass

$$\textbf{TIME}(f(n)) = \textbf{TIME}(2^{f(n)})$$

(Wh: eine rekursive Funktion ist eine Funktion, für die es eine TM gibt, die diese Funktion berechnet).