Komplexitätstheorie

$\mathsf{P} = \mathsf{NP}$?

private Mitschrift – *kein* offizielles Script – www.kuertz.net/uni

`Id: Komplexitaet.tex,v 1.15 2005/07/04 10:11:01 mtu Exp`

Disclaimer: Dies ist kein offizielles Script, sondern nur eine private Mitschrift. Ich kann daher keine Gewähr für die Richtigkeit übernehmen. Vor allem weicht die Nummerierung der Kapitel und Sätze zum Teil von der in der Vorlesung verwendeten ab. Falls jemand einen Fehler entdeckt, so möge er/sie mir bitte eine eMail schicken - vielen Dank!

Max Tuengerthal (max@17q.de)
und Klaas Ole Kürtz (uni@kuertz.net)

Inhaltsverzeichnis

1 Einführung

In der Komplexitätstheorie beschäftigt man sich mit der **Schwierigkeit von Berechnungsproblemen** bezüglich gewisser **Ressourcen** (Zeit, Speicherplatz etc.). Dabei konzentriert man sich auf **Klassen** von Problemen, die die gleichen Ressourcen benötigen (P, NP, ...) und deren Beziehungen. Es geht nicht um die Analyse **einzelner** Algorithmen – hier sei z.B. auf die Vorlesung *Effiziente Algorithmen* verwiesen.

1.1 Das Erreichbarkeitsproblem

Gegeben sei ein gerichteter Graph $G = (V, E)$ mit einer endlichen Menge von Knoten V und einer endlichen Menge von Kanten E, gegeben seien weiter $a, b \in V$. Das *Erreichbarkeitsproblem* läßt sich als Entscheidungsproblem formulieren, ob es einen Pfad in G von a nach b gibt.

Ein Algorithmus, der das Problem mit Eingabe $G = (V, E); a, b \in V$ löst:

```
S = { a };
M = { a };
while (S ≠ ∅)
  u ∈ S;
  S = S \ { u };
  for all (u, v) ∈ E, v ∉ M
    M = M ∪ { v };
    S = S ∪ { v };
  if (b ∈ M) return "yes";
return "no";
```

Diese Beschreibung des Algorithmus' ist (bewußt) **unpräzise**, es stellen sich u.a. folgende Fragen:

- In welcher Form ist die Eingabe gegeben?
- Wie wird $u \in S$ gewählt (FIFO entspräche Breitensuche, LIFO entspricht Tiefensuche, Kombinationen etc.)?

Die unpräzise Beschreibung des Algorithmus' reicht aber aus, da die Einzelheiten nicht immer interessieren: Wichtig ist neben der **Korrektheit** des Algorithmus' die **Effizienz** des Algorithmus' für jede (sinnvolle) Interpretation.

In diesem Fall läßt sich die Laufzeit grob mit $\mathcal{O}(|E|)$ angeben.

1.2 $\mathcal{O}$-, Θ-, Ω-Notation

Definition: Seien $f, g\colon N \to R^{\geq 0}$. Dann gilt:

$$f \in \mathcal{O}(g) :\Longleftrightarrow \exists\, c > 0\, \exists\, n_0 \in \mathbb{N}\, \forall\, n \geq n_0 \colon f(n) \leq c \cdot g(n)$$
$$f \in \Omega(g) :\Longleftrightarrow g \in \mathcal{O}(f)$$
$$f \in \Theta(g) :\Longleftrightarrow f \in \mathcal{O}(g) \cap \Omega(g)$$

Beispiele:

1. Sei $p(n)$ ein Polynom d-ten Grades, dann ist $p(n) \in \mathcal{O}(n^d)$.
2. Für die Fakultät gilt:

$$n! = 2^{\log n + \log(n-1) + \ldots + \log 1} \leq 2^{n \cdot \log n} \in 2^{\mathcal{O}(n \log n)}$$

   Es gilt aber auch: $n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1$, also besteht $n!$ aus $\frac{n}{2}$ Faktoren der Form $(n-i)(i+1) \geq n$ für $i \leq \frac{n}{2}$, damit gilt $n! \geq n^{\frac{n}{2}} = 2^{\frac{n}{2} \log n} \in 2^{\Theta(n \log n)}$.

   Damit gilt $n! \in 2^{\Omega(n \log n)}$.

Definition: *Polynomzeit-Algorithmen* sind Algorithmen, die für Eingaben der Länge n höchstens $\mathcal{O}(n^d)$ Schritte benötigen, wobei d fest ist.

Dies sind die Algorithmen mit „*akzeptabler*“ Zeitkomplexität – nicht mehr akzeptabel wären Algorithmen mit exponentieller Laufzeit etc. Damit sind die Polynomzeit-Algorithmen eine theoretische Annäherung an den **intuitiven** Begriff „akzeptabel“. **Kritik** daran:

- Für große d ($\mathcal{O}(n^{100})$) oder große Konstanten ($2^{100000} \cdot n$) ist in der Praxis ein theoretisch exponentieller Algorithmus (z.B. mit $2^{\frac{n}{100}}$ oder $n^{\log \log n}$) besser! Dieser Fall taucht jedoch eher selten auf.
- Man betrachtet die **Worst-Case-Komplexität**, in manchen Fällen ist aber eine Analyse der mittleren Laufzeit besser (siehe z.B. QuickSort). Hier ist es jedoch z.T. schwer, die Verteilung der Eingabe zu bestimmen (Was sind „typische“ Eingaben?).

Die theoretische Annäherung ist dennoch **robust** in dem Sinne, daß man nicht von einem bestimmten Maschinenmodell abhängt.

Bei der **Platzkomplexität** mißt man den Speicherplatz für die **Daten zur Laufzeit**, dies schließt den Platz für die Eingabe und für den Programmcode üblicherweise aus (Ausnahme: Schaltkreiskomplexität).

1.3 Das Problem MAXFLOW

Gegeben sei ein *Netzwerk* $N = (V, E, s, t, c)$. Dabei ist $G = (V, E)$ ein gerichteter Graph mit $(u, v) \in E \Rightarrow (v, u) \notin E$ $(\star)$. $s \in V$ ist eine *Quelle* (nur ausgehende Kanten), $t \in V$ eine *Senke* (nur eingehende Kanten), und $c\colon E \to \mathbb{N}$ ist eine Kapazitätsfunktion.

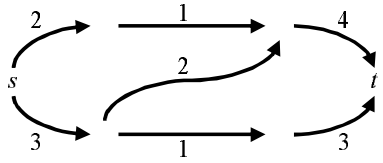
Definitionen: Ein *Fluß* ist eine Funktion $f\colon E \to \mathbb{N}$, so daß

1. $f(u, v) \leq c(u, v)$
2. $\forall v \notin \{s, t\}\colon \sum_{u\colon (u,v)\in E} f(u, v) = \sum_{w\colon (v,w)\in E} f(v, w)$.

Der *Wert* eines Flußes $W(f)$ ist definiert als

$$W(f) = \sum_{(s,v)\in E} f(s, v) = \sum_{(v,t)\in E} f(v, t)$$

Beispiel:



Nun läßt sich ein **Optimierungsproblem** definieren:

MAXFLOW ist das Problem, das zu einem gegebenen Netzwerk N einen Fluß f mit maximalem $W(f)$ findet.

Das zugehörige **Entscheidungsproblem**:

MAXFLOW(D) ist das Problem, ob es zu einem gegebenen Netzwerk N einen Fluß f mit $W(f) \geq k$ findet.

Falls man MAXFLOW in polynomieller Zeit lösen kann, so auch MAXFLOW(D); es gilt auch die Umkehrung (nicht so trivial).

Wir prüfen zunächst, ob ein Fluß f optimal ist. Sei f' ein Fluß mit $W(f') > W(f)$. Wir definieren dann zunächst $\Delta f\colon E \to \mathbb{Z}$ (kantenweise) durch $\Delta f = f' - f$; dann gilt $-f \leq \Delta f \leq c - f$.

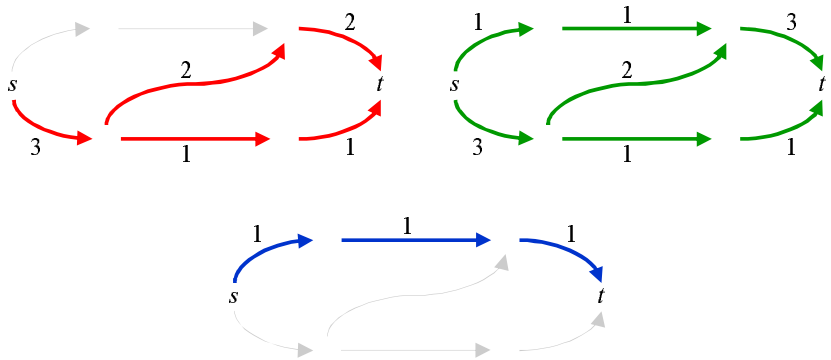
Damit ist Δf ein Fluß in N mit positivem Wert, aber mit negativen Flüssen. Somit ist aber Δf ein Fluß in einem neuen Netzwerk $N(f) = (V, E', s, t, c')$ mit $E' = E \cup E^{-1}$, wobei $E^{-1} = \{(v,u) \mid (u,v) \in E\}$ ist (nun ist die Bedingung $(\star)$ nicht mehr erfüllt!). Es sei

$$\begin{aligned} c'(u,v) &= c(u,v) - f(u,v) && \text{für } (u,v) \in E \\ c'(u,v) &= f(v,u) && \text{für } (u,v) \in E^{-1} \end{aligned}$$

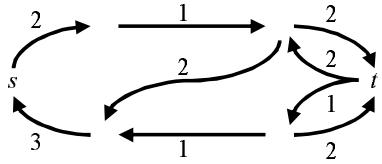
Nun ist Δf ein Fluß in $N(f)$, wenn wir setzen:

- für alle (u,v) mit $f'(u,v) < f(u,v)$ sei $\Delta f(v,u) = f(u,v) - f'(u,v)$ und $\Delta f(u,v) = 0$
- für alle (u,v) mit $f'(u,v) \geq f(u,v)$ sei $\Delta f(u,v) = f'(u,v) - f(u,v)$ und $\Delta f(v,u) = 0$

Beispiel: Betrachte wieder obiges Beispiel mit einem Fluß f links und einem besseren Fluß f' rechts sowie der Differenz $\Delta f = f' - f$ darunter:



Als neues Netzwerk $N(f)$ für Δf ergibt sich nun:



Man erkennt am Beispiel: f ist optimal genau dann, wenn es keinen Fluß Δf in $N(f)$ mit $W(\Delta f) > 0$ gibt. Dies ist genau dann der Fall, wenn es keinen Pfad von s nach t in folgendem Graphen gibt:

$$(V, \{\, (u,v) \in E' \mid c'(u,v) > 0\}$$

Damit ist das Problem auf das Erreichbarkeitsproblem zurückgeführt; man kann also in polynomieller Zeit feststellen, ob ein gegebener Fluß optimal ist.

Satz: Der Fluss von f ist das, was in die Senke fließt, d.h.

$$W(f) = \sum_{(v,t)\in E} f(v,t)$$

Beweis: Sei $S \subseteq V \setminus \{t\}$ mit $s \in S$.

$$W(f) = \sum_{i \in S} \Big(\underbrace{\sum_{j:\,(i,j)\in E} f(i,j)}_{(*)} - \underbrace{\sum_{j:\,(j,i)\in E} f(j,i)}_{(**)} \Big)$$

Falls $i \neq s$, dann gilt: $(*) - (**) = 0$.
Falls $i = s$, dann gilt: $(*) = W(f)$ und $(**) = 0$.

$$W(f) = \sum_{(i,j)\in (S\times\overline{S})\cap E} f(i,j) - \sum_{(i,j)\in (\overline{S}\times S)\cap E} f(i,j)$$

Wir haben Kanten $(a,b) \in (S \times S) \cap E$ weggelassen, aber diese tauchen in $(*)$ *und* in $(**)$ auf, d.h. sie heben sich auf.

Für $S = V \setminus \{t\}$ erhalten wir

$$W(f) = \sum_{(v,t)\in E} f(v,t).$$

□

Definition: Der Wert eines Flusses kann allgemeiner definiert werden:

$$W(f) := \sum_{v:(s,v)\in E} f(s,v) - \sum_{v:(v,s)\in E} f(v,s)$$

Definition: Für einen Fluss f definiere

$$N(f) := (V, E', s, t, c')$$

mit

$$\begin{aligned} E' &:= E \cup E^{-1} \\ c'(u,v) &:= c(u,v) - f(u,v) \text{ für alle } (u,v) \in E \\ c'(v,u) &:= f(u,v) \text{ für alle } (v,u) \in E^{-1} \end{aligned}$$

Satz:
f optimal $\Leftrightarrow$ es gibt keinen Fluss Δf in $N(f)$ mit $W(\Delta f) > 0$
$\Leftrightarrow$ es gibt in (V, E') keinen Pfad von s nach t entlang Kanten (u, v) mit $c'(u, v) > 0$.

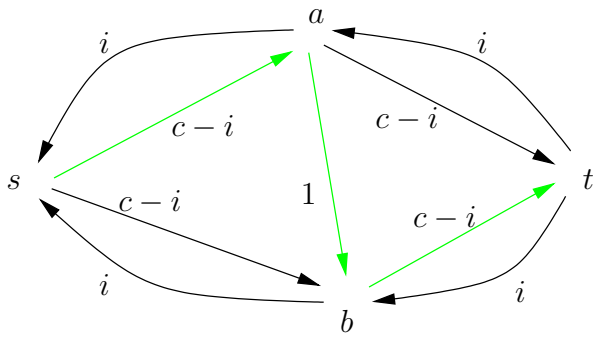
Beweis: Übung.

Ford-Fulkerson-Algorithmus für MAXFLOW mit Eingabe $N = (V, E, s, t, c)$:

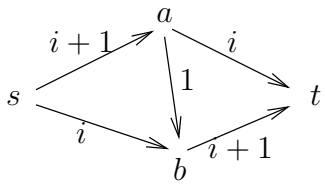
```
f = 0;
while () {
  konstruiere N(f);
  // Erreichbarkeitsproblem:
  suche Pfad P (ohne Zykel)
       von s nach t in N(f) über
       Kanten (u, v) mit c'(u,v) > 0
  setze c_0 als die kleinste Kapazit\"at in P
  for all ((u, v) ∈ E) {
    if ((u, v) ∈ E ∩ P)
      f(u, v) = f(u, v) + c_0;
    if ((v, u) ∈ E^-1 ∩ P)
      f(u, v) = f(u, v) - c_0;
  }
}
```

Komplexität: Sei n die Anzahl der Knoten und $C = \max\{c(u,v) \mid (u,v) \in E\}$. Dann ist die Komplexität für eine Iteration in $\mathcal{O}(n^2 \log C)$, und die Anzahl der Iterationen ist $\leq n \cdot C$. Insgesamt ergibt sich:

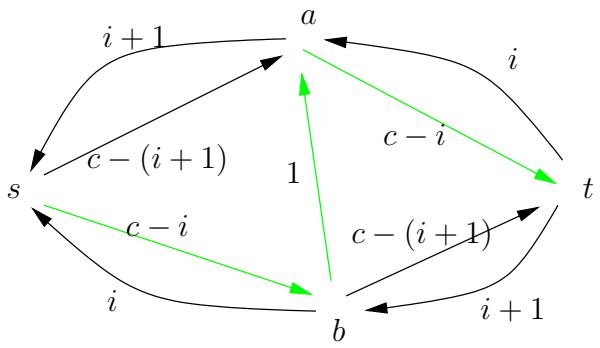
$$\mathcal{O}(n^3 \cdot C \cdot \log C)$$



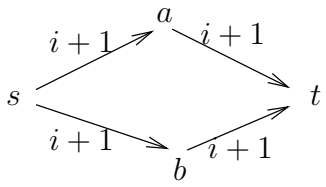
$N(f_{2i})$



f_{2i+1}



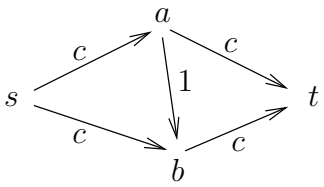
$N(f_{2i+1})$



$f_{2(i+1)}$

Problem: C ist exponentiell groß in der Eingabe. Also ist dies kein Polynomzeit-Algolrithmus!

Beispiel: Sei folgendes Netz gegeben:



Im Worst-Case benötigt der Ford-Fulkerson-Algorithmus $2c$ Iterationen. **Annahme**: Der Suchalgorithmus findet immer Pfade, die (a, b) oder (b, a) enthalten. Dann werden die Flüsse $f_0, f_1, \ldots, f_{2c}$ berechnet mit $f_0 = 0$, $f_{2c} = f_{opt}$.

Verbesserung (Edmonds-Karp-Algorithmus): Implementierung der Suche, so dass kürzeste Pfade gefunden werden (Breitensuche).

Definition: Eine Kante in P mit minimaler Kapazität heisst *Flaschenhals.*

Satz: Wenn immer kürzester Pfad P gewählt wird, dann ist jede Kante $(u, v) \in E$ oder ihre Umkehrung in höchstens n Iterationen ein Flaschenhals.

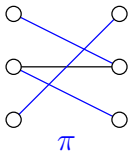
Beweis: Übung.

Damit ist die Anzahl der Iterationen $\leq n^3$, also ist die Laufzeit des Edmonds-Karn-Algorithmus' in $\mathcal{O}(n^5 \cdot \log C)$. D.h. MAXFLOW kann in polynomieller Zeit gelöst werden.

1.4 Das Heiratsproblem (MATCHING)

Heiratsproblem (MATCHING) ist das Problem, zu einem bipartiten Graph $B = (U, V, E)$ mit $|U| = |V| = n$, $E \subseteq U \times V$ eine bijektive Abbildung $\pi \colon U \to V$ zu finden, so dass $(u, \pi(u)) \in E$ ist für alle $u \in U$.

Beispiel:



Definition: Sei $\pi : U \to V$ bijektiv. Ein *(bipartites) Matching* ist dann

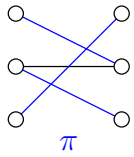
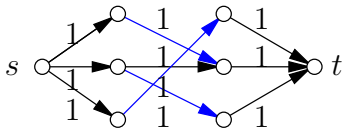
$$M := \{(u, \pi(u)) \mid u \in U\} \subseteq E$$

Reduktion auf MAXFLOW(D) (Entscheidungsproblem): Zu jedem bipartiten Graphen $B = (U, V, E)$ wie oben konstruieren wir ein Netzwerk N, so dass gilt:

$$B \text{ erlaubt bipartites Matching} \iff \underbrace{N \text{ hat Fluss } f \text{ mit } W(f) \geq n}_{\text{Instanz von MAXFLOW(D)}} \qquad (\star)$$

Es folgt, dass MATCHING polynomiell ist (falls die Reduktion polynomiell ist).

Idee der Reduktion:

Aus  wird 

Definiere nun das Netzwerk N:

$$N := (U \cup V \cup \{s,t\}, E \cup \{(s,u) \mid u \in U\} \cup \{(v,z) \mid v \in V\}, s, t, c)$$
$$\text{mit } c(u,v) := 1 \quad \text{für alle Kanten in } N$$

Man sieht leicht, dass $(\star)$ gilt. Damit ist MATCHING in polynomieller Zeit entscheidbar.

Notation:
$A \leq B$: A ist auf B reduzierbar.

Satz: Es gilt
- Falls $A \leq B$ und B „leicht“ ist, so ist auch A „leicht“.
- Falls $A \leq B$ und A „schwer“ ist, so ist auch B „schwer“.

1.5 Das Travelling-Salesman-Problem (TSP)

Das **Travelling-Salesman-Problem** (TSP) ist das Optimierungsproblem, zu $d\colon \{1,\ldots,n\} \times \{1,\ldots,n\} \to \mathbb{N}$ (Distanzmatrix) die kürzeste Tour zu finden, d.h. eine Permutation π auf $\{1,\ldots,n\}$, so dass folgendes minimal ist:

$$\sum_{i=1}^{n-1} d(\pi(i), \pi(i+1)) + d(\pi(n), \pi(1)).$$

Das Problem TSP(D) ist das zugehörige Entscheidungsproblem, d.h. entscheide, ob es zu gegebenem d (wie oben) und k ein π (wie oben) gibt, so dass die obige Summe $\leq k$ ist.

Das Travelling-Salesman-Problem ist ein **NP-vollständiges Problem**! Ein trivialer Algorithmus probiert alle möglichen Touren aus, das sind aber $(n-1)!$ Touren! Ein Polynomzeit-Algorithmus ist nicht bekannt.

2 Turingmaschinen und Berechenbarkeit

2.1 Die 1-Band-TM: Ein universelles Berechnungsmodell

Definition: Eine 1-Band-TM (ohne E/A-Band) ist ein Tupel

$$M = (Q, \Sigma, \Gamma, q_0, F.\delta)$$

Q – endl. Zustandsmenge
Σ – Arbeitsalphabet, $\{0, 1, \square, \#\} \subseteq \Sigma$, $\#$ – Trennsymbol, $\square$ – blank
Γ – Eingabealphabet
$q_0 \in Q$ – Anfangszustand
$F \subseteq Q$ – Endzustandsmenge
$\delta\colon (Q \setminus F) \times \Sigma \to Q \times \Sigma \times \{-1, 0, 1\}$ – (deterministische) partielle Übergangsfunktion

Definition: Eine *Konfiguration* einer TM M ist ein Tupel

$$C = (q, w, p) \in Q \times \sigma^* \times \mathbb{N}$$

q – aktueller Zustand
p – aktuelle Position
$w = w_0 \dots w_{n-1} \in \Sigma^*$ definiert unendliche Bandbeschriftung, $\hat{w}\colon \mathbb{N} \to \Sigma$,

$$\hat{w}(i) = \begin{cases} w_i & i < n \\ \square & i \geq n \end{cases}$$

Wir identifizieren w und $\hat{w}$.
$\mathcal{C}$ – Menge der Konfigurationen von M
δ wird erweitert zu partieller Übergangsfunktion

$$\Delta\colon \mathcal{C} \to \mathcal{C},$$

mit

$$C = (q, w, p),\ w(p) = a \in \Sigma,$$

$$\delta(q, a) = (q', b, d),$$

$$\Delta(C) = \begin{cases} (q', w', p + d) & p + d \geq 0 \\ (q', w', 0) & p + d < 0 \end{cases}$$

mit

$$w'(i) = \begin{cases} w(i) & i \neq p \\ b & i = p \end{cases}$$

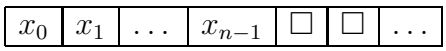
$\Delta(C)$ ist die *Nachfolgekonfiguration* von C.

Notation: $C \vdash_M C'$ mit $C' = \Delta(C)$.

Definition: C' ist erreichbar von C ($C \vdash_M^* C'$), wenn eine Folge $C_0, C_1, \ldots, C_t$ existiert mit $C_0 = C$, $C_t = C'$ und $C_i \vdash_M C_{i+1}$ für alle $i < t$.

Anfangskonfiguration von M auf x ist

$$C_0(x) = (q_0, x, 0)$$



Endkonfiguration:

$$C = (q, w, p) \text{ mit } q \in F$$

Definition: Eine *Berechnung* von M auf x ist eine Folge $C_0, C_1, \ldots$ von Konfigurationen, so dass $C_0 = C_0(x)$, $C_i \vdash_M C_{i+1}$, für alle $i \geq 0$.
Die Berechnung ist *vollständig*, wenn sie entweder unendlich ist oder in einer Endkonfiguration endet. Im letzteren Fall *hält* M auf x.

Bemerkung: Man kann jede TM in eine überführen, die immer eine vollständige Berechnung liefert.

2.2 Berechenbarkeit, Aufzählbarkeit, Entscheidbarkeit

Definition: Für eine TM M definieren wir

$$L(M) := \{x \in \Gamma^* \mid M \text{ hält auf } x\}.$$

$L \subseteq \Gamma^*$ ist *rekursiv aufzählbar* gdw. eine TM M existiert mit $L(M) = L$.

Definition: $C_0, \ldots, C_t$ sei eine vollständige Berechnung von M auf x mit Endkonfiguration $C_t = (q, y, p)$, wobei $y = y_0 \ldots y_m \in \Gamma^*$ (d.h. insb. $y_i \neq \square$, $i \in \{0, \ldots, m\}$), dann nennen wir y *das Ergebnis der Berechnung von M auf x*.

M *berechnet* also eine partielle Funktion $f_M \colon \Gamma^* \to \Gamma^*$ mit Definitionsbereich $L(M)$.
Eine partielle Funktion $f \colon \Gamma^* \to \Gamma^*$ ist turing-berechenbar, wenn eine (1-Band-) TM M existiert mit $f_M = f$.

Definition: Ein (1-Band-) *Akzeptor* M ist eine 1-Band-TM, deren Menge F von Endzuständen zerlegt ist in eine Menge F^+ von akzeptierenden Zuständen und F^- von verwerfenden Zuständen.
M *akzeptiert* x, falls M hält auf x in einem Zustand aus F^+.
M *verwirft* x, falls M hält auf x in einem Zustand aus F^-.

Definition: Sei $L \subseteq \Gamma^*$. M *entscheidet* L, wenn für alle $x \in \Gamma^*$ gilt,

$$x \in L \Rightarrow M \text{ akzeptiert } x$$

$$x \notin L \Rightarrow M \text{ verwirft } x$$

(insbesondere $L(M) = \Gamma^*$)
$L \subseteq Gamma^*$ ist *entscheidbar* (oder *rekursiv*), falls es einen Akzeptor M gibt, welcher L entscheided.

Eine äquivalente Definition wäre:
$L \subseteq \Gamma^*$ ist entscheidbar gdw. die Charakteristische-Funktion $\chi \colon \Gamma^* \to \{0,1\}$ Turingberechenbar ist.

Fakt: (Grundstudium)

- $L \subseteq \Gamma^*$ ist entscheidbar gdw. L und $\Gamma^* \setminus L$ rekursiv aufzählbar sind.
- Sei $L \subseteq \Gamma^*$, $L \neq \emptyset$. Dann gilt:
  L rekursiv aufzählbar $\Leftrightarrow$ es gibt Turingberechnebare totale Funktion $f \colon \mathbb{N} \to \Gamma^*$ mit $\text{Bild}(f) = L$.

Kodierung σ:

$$\text{TM } M \mapsto \sigma(M) \in \{0,1\}^*$$

$$x \in \Gamma^* \mapsto \sigma(x) \in \{0,1\}^*$$

Fakt: (Grundstudium)
Man kann eine *universelle* TM U konstruieren, die auf Eingabe $\sigma(M)\#\sigma(x)$ die Berechnung von M auf x nachvollzieht und hält gdw. M auf x hält. (U ist Interpreter für TM)

Halteproblem für TM

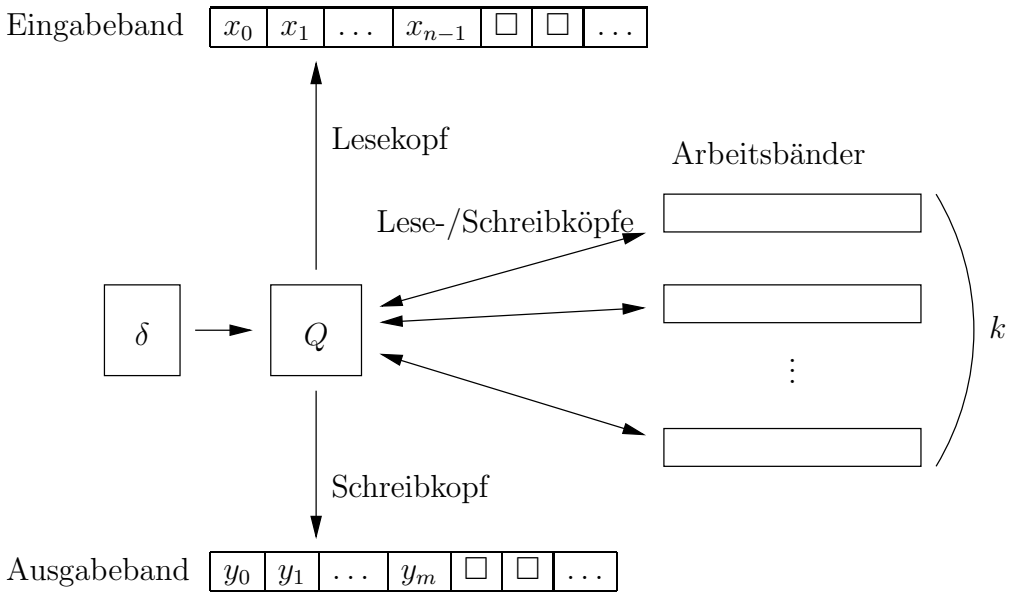
$$H = (\{\sigma(M)\#\sigma(x) \mid x \in L(M),\ M \text{ TM}\} = L(U))$$

Die folgenden Probleme sind unentscheidbar:

- H
- $H_0 = \{\sigma(M) \mid \sigma(M) \in L(M),\ M \text{ TM}\}$
- $H_\epsilon = \{\sigma(M) \mid \epsilon \in L(M),\ M \text{ TM}\}$
- $H_A = \{\sigma(M)\#\sigma(M') \mid M \text{ und } M' \text{ sind äquivalent, d.h. } L(M) = L(M')\}$

Siehe auch Satz von Rice aus dem Grundstudium.

2.3 Allgemeinere Modelle von TM



Definition: Eine TM mit Ein- und Ausgabeband (E/A-Band) und k Arbeitsbändern ist gegeben durch

$$M = (Q, \Gamma_E, \Gamma_A, \Sigma, q_0, F, \delta)$$

Q – wie immer
Γ_E – Alphabet für Eingabe
Γ_A – Ausgabealphabet, $\Gamma_A \subseteq \Sigma \setminus \{\square\}$
Σ – Arbeitsalphabet

q_0 – wie immer
F – wie immer
$\delta\colon (Q \setminus F) \times \Gamma_E \times \Sigma^k \to Q \times \{-1,0,1\} \times \Sigma^k \times \{-1,0,1\}^k \times (\Gamma_A \cup \{*\})$. ($*$ steht für keine Ausgabe)

Definition: Eine (partielle) *Konfiguration* von M ist ein Tupel

$$C = (q, w_1, \ldots, w_k, p_1, \ldots, p_k, p_E, p_A) \in Q \times (\Sigma^*)^k \times \mathbb{N}^{k+2}$$

q – aktueller Zustand
w_i – Bandbeschriftung des i-ten Arbeitsbandes
p_i – Position des Schreib-/Lesekopfes des i-ten Arbeitsbandes
p_E – Position des Lesekopfes des E-Bandes
p_A – Position des Schreibkopfes ds Ausgabebandes
Totale Konfiguration: partielle Konfiguration mit Beschriftung der E/A-Bänder

$$(q, w_1, \ldots, w_k, w_E, w_A, p_1, \ldots, p_k, p_E, p_A)$$

Totale Ausgangskonfiguration auf $x \in \Gamma_E^*$:

$$C_0(x) = (q_0, \underbrace{\epsilon, \ldots, \epsilon}_{k}, x, \epsilon, \underbrace{0, \ldots, 0}_{k}, 0, 0)$$

Eine *Berechnung* einer k-Band TM mit E/A-Band ist eine Folge von totalen Konfigurationen angefangen mit der Ausgangskonfiguration und gemäß δ.

Varianten dieses Modells:

- kein seperates E/A-Band (k-Band TM ohne E/A-Band).
- Einweg-Eingabeband.
- TM mit d Speicherzellen, die natürliche Zahlen aufnehmen (d fest gegeben). In jedem Schritt kann die Maschine 1 zu einer Zelle addieren oder subtrahieren.
- TM mit mehreren Köpfen auf einem Band.
- Baum-TM.

Häufig sind unsere Komplexitätsklassen unabhängig von speziellen Modellen. Unterschiede können sich für kleinere Komplexitätsklassen bzw. genauere Analysen ergeben.

Unser Standardmodell wird eine k-Band TM ohne E/A-Band sein.

3 Komplexitätsklassen

Ab jetzt: TM ist immer ein k-Band-Akzeptor. Wenn ein seperates Ein- oder Ausgabeband vorhanden ist, wird dieses bei den k Bändern mitgezählt.

Definition: Sei M eine TM und x eine Eingabe für M.
- $\text{time}_M\, x :=$ Länge der vollständigen Berechnungen von M auf x.
- $\text{space}_M\, x :=$ Anzahl der im Verlauf der Berechnung besuchten Zellen der Bänder (ohne Ein- und Ausgabeband).

Offensichtlich: $k \cdot \text{time}_M(x) \geq \text{space}_M(x)$.

Seien $T, S\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$. Eine TM M heißt T-zeitbeschränkt bzw. S-platzbeschränkt, falls für alle Eingaben $x \in \Gamma^*$ gilt:
- $\text{time}_M(x) \leq T(|x|)$ bzw.
- $\text{space}_M(x) \leq S(|x|)$.

Seien $T, S\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$.

$$\begin{aligned}
\mathsf{DTIME}_k(T) &:= \{L \subseteq \Gamma^* \mid \text{es gibt } T\text{-zeitbeschränkte } k\text{-Band TM } M, \text{ welche } L \text{ entscheidet}\}.\\
\mathsf{DTIME}(T) &:= \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}_k(T).\\
\mathsf{DSPACE}_k(S) &:= \{L \subseteq \Gamma^* \mid \text{es gibt } S\text{-platzbeschränkte } k\text{-Band TM } M, \text{ welche } L \text{ entscheidet}\}.\\
\mathsf{DSPACE}(S) &:= \bigcup_{k\in\mathbb{N}} \mathsf{DSPACE}_k(S).\\
\mathsf{DTIME\text{-}SPACE}_k(T,S) &:= \{L \subseteq \Gamma^* \mid \text{es gibt } T\text{-zeitbeschränkte und } S\text{-platzbeschränkte } k\text{-Band TM } M, \text{ welche } L \text{ entscheidet}\}.\\
\mathsf{DTIME\text{-}SPACE}(T,S) &:= \bigcup_{k\in\mathbb{N}} \mathsf{DTIME\text{-}SPACE}_k(T,S).
\end{aligned}$$

Satz (Bandreduktion): Sei $S(n) \geq n$. Dann ist

$$\mathsf{DTIME\text{-}SPACE}(T,S) \subseteq \mathsf{DTIME\text{-}SPACE}_1(\mathcal{O}(T(n)\cdot S(n)), S(n)).$$

Beweis: Übung: Serie 3, Aufgabe 2.

Bemerkung: Bei einer 1-Band TM wird die Eingabe beim Platzverbrauch mitgezählt. Deshalb $S(n) \geq n$.

KOROLLAR:
$$\mathsf{DTIME}(T) \subseteq \mathsf{DTIME}_1(\mathcal{O}(T^2)).$$

Beweis: Vorheriger Satz und $\text{space}_M(x) \leq \text{time}_M(x) \cdot k$. □

KOROLLAR: Für $S(n) \geq n$ gilt
$$\mathsf{DSPACE}(S) \subseteq \mathsf{DSPACE}_1(S).$$

Satz:
a) Für alle T (Zeitfunktion) gilt: $\mathsf{DTIME}(T) \subseteq \mathsf{DSPACE}(T)$.
b) Für $S(n) \geq \log n$ gilt: $\mathsf{DSPACE}(S) \subseteq \mathsf{DTIME}(2^{\mathcal{O}(S)})$.

Beweis:

a) Ist klar.

b) Sei $L \in \mathsf{DSPACE}(S)$, M sei TM mit Eingabeband und k Arbeitsbändern, welche L mit Platz $S(n)$ entscheidet. Für jede Eingabe x gilt:
In der Berechnung von M auf x kommt keine partielle Konfiguration mehr als einmal vor. (partielle Konfiguration ist vollständige Konfiguration, ohne Beschreibung der Ein- und Ausgabebänder). Sonst würde M in eine unendliche Schleife geraten, und somit nicht halten (*beachte:* M ist deterministisch). Anzahl der partiellen Konfigurationen mit Platz $S(n)$ ist beschränkt durch:
$$|Q| \cdot (n+1) \cdot S(n)^k \cdot |\Sigma|^{S(n)} = 2^{\mathcal{O}(S(n))}.$$
Also ist $\text{time}_M(x) = 2^{\mathcal{O}(S(n))}$. □

Definition: Definiere
- $\mathsf{LOGSPACE} := \bigcup_{k \in \mathbb{N}} \mathsf{DSPACE}(k \cdot \log n)$ und
- $\mathrm{P} := \bigcup_{k \in \mathbb{N}} \mathsf{DTIME}(n^k)$.

KOROLLAR: Es gilt $\mathsf{LOGSPACE} \subseteq \mathrm{P}$.

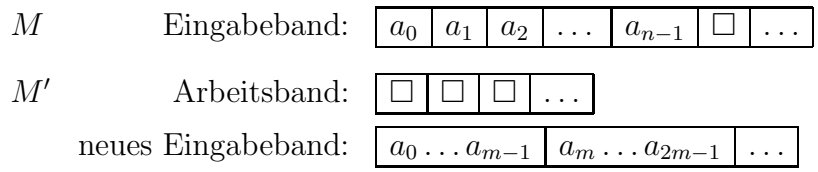
3.1 Speedup und Bandkompression

Satz (Das lineare speedup-Theorem): Sei $k > 1$ und $\epsilon > 0$. Dann gilt für alle $T\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ mit $n = o(T(n))$ (d.h. $\lim_{n\to\infty} \frac{n}{T(n)} = 0$), dass

$$\mathsf{DTIME}_k(T(n)) \subseteq \mathsf{DTIME}_k(\max\{n, \epsilon \cdot T(n)\}).$$

Beweis: Sei M eine k-Band-TM, welche L in Zeit $T(n)$ entscheidet. Sei $m \in \mathbb{N}$, so dass $\epsilon \cdot m \geq 16$. Sei Σ das Arbeitsalphabet von M.
Sei M' k-Band-TM mit Alphabet $\Sigma^m \cup \Sigma$. Beschreibung von M':

Phase 1: M' kopiert die Eingabe auf ein anderes Band, wobei m Symbole in eines komprimiert werden.



Auf dem neuen Eingabeband sind dann $\lceil \frac{n}{m} \rceil$ Symbole. Die Anzahl der Schritte, die in Phase 1 ausgeführt werden, ist $n + \lceil \frac{n}{m} \rceil$. (Lesen der Eingabe und Zurücksetzen des Kopfes auf dem neuen Eingabeband).

Phase 2: M' simuliert jeweils m Schritte von M in acht Schritten wie folgt:

a) M' speichert in der Zustandsmenge den Inhalt der beiden Nachbarfelder des gerade bearbeiteten Feldes. Dazu sind vier Schritte nötig.

b) M' bestimmt das Resultat der nächsten m Schritte von M (fest in der Übergangsfunktion von M' kodiert), denn in m Schritten kann M nur den Inhalt von Feldern benutzen bzw. verändern, welche höchstens m Schritte entfernt sind.

M' braucht weitere vier Schritte, um die entsprechenden Änderungen durchzuführen.

M' merkt sich im Zustandsraum, an welcher Stelle innerhalb eines Symols in Σ^m die Köpfe von M stehen.

c) M' akzeptiert bzw. verwirft genau dann, wenn M akzeptiert bzw. verwirft.

Zeitabschätzung: Sei x die Eingabe der Länge n.

$$\text{time}_M(x) \leq n + \lceil \frac{n}{m} \rceil + \lceil 8\frac{T(n)}{m} \rceil \leq n + \frac{n}{m} + 8\frac{T(n)}{m} + 2.$$

Da $n = o(T(n))$, gibt es zu jedem $d > 0$ ein n_d, so dass

$$\frac{T(n)}{n} \geq d$$

für alle $n \geq n_d$. Also: $n \leq \frac{T(n)}{d}$.

Sei $n \geq \max\{2, n_d\}$. Dann gilt $2n \geq n + 2$. Es gilt

$$\begin{aligned}
\text{time}_{M'}(x) &\leq 2n + \frac{n}{m} + 8\frac{T(n)}{m} \\
&\leq T(n)\left(\frac{2}{d} + \frac{1}{m \cdot d} + \frac{8}{m}\right) \\
&= T(n)\left(\frac{2m + 8d + 1}{m \cdot d}\right)
\end{aligned}$$

für $|x| = n \geq \max\{2, n_d\}$. Setze $d := \frac{2m+1}{8}$. Dann ist

$$\begin{aligned}
\text{time}_{M'}(x) &\leq T(n)\left(\frac{8((2m+1) + 2m + 1)}{m(2m+1)}\right) \\
&= \frac{16}{m} \cdot T(n) \leq \epsilon \cdot T(n)
\end{aligned}$$

für alle $|x| = n \geq \max\{2, n_d\}$.

Die endlich vielen Eingaben der Länge $< n_d$ können in Zeit n entschieden werden.

Es folgt $L \in \mathsf{DTIME}_k(\max\{n, \epsilon \cdot T(n)\})$. □

KOROLLAR: Sei $\epsilon > 0$. Dann gilt für alle $T\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ mit $n = o(T(n))$, dass

$$\mathsf{DTIME}(T(n)) \subseteq \mathsf{DTIME}(\max\{n, \epsilon \cdot T(n)\}).$$

Satz (Bandkompression): Für alle $S\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$, $k > 1$ und alle $\epsilon > 0$ gilt

$$\mathsf{DSPACE}_k(S(n)) \subseteq \mathsf{DSPACE}(\max\{1, \epsilon \cdot S(n)\}).$$

Beweis: Analog, aber einfacher!

3.2 Das GAP-Theorem

Frage: Wenn S_2 schneller wächst als S_1, ist dann $\mathsf{DSPACE}(S_2) \supsetneq \mathsf{DSPACE}(S_1)$? Analog für Zeit.

Das GAP-Theorem sagt: "Im Allgemeinen nicht!"

Lemma: Sei M eine TM mit

$$\max_{|x|=n}(space_M(x)) \leq S(n)$$

für "fast alle" (alle bis auf endlich viele) $n \in \mathbb{N}$.
Dann ist $E(M) \in \mathsf{DSPACE}(S(n))$.
$E(M) := \{x \mid M \text{ hält auf } x \text{ mit akzeptierendem Zustand}\}$.

Beweis: Die Menge $X = \{x \mid space_M(x) > S(|x|)\}$ ist nach Vorr. endlich, also können wir $\chi_{E(M)}(x)$ für $x \in X$ fest in der Übergangsfunktion von M kodieren und ohne Benutzung des Speichers entscheiden.

Für Eingabe $x \in X$ hat man Speicherverbrauch 0. Die restlichen Eingaben $x \notin X$ werden durch Simulation von M entschieden. □

Borodin's **GAP-Theorem:** Für jede totale und berechenbare Funktion $g\colon \mathbb{N} \to \mathbb{N}$ mit $g(n) \geq n$ gibt es eine totale und berechenbare Funktion $S\colon \mathbb{N} \to \mathbb{N}$, so dass

$$\mathsf{DSPACE}(S) = \mathsf{DSPACE}(g \circ S).$$

Folgerung: Es gibt Funktionen S_1, S_2 und S_3 mit

- $$\mathsf{DSPACE}(S_1) = \mathsf{DSPACE}(2^{S_1})$$
- $$\mathsf{DSPACE}(S_2) = \mathsf{DSPACE}(2^{2^{S_1}})$$
- $$\mathsf{DSPACE}(S_3) = \mathsf{DSPACE}(2^{2^{\cdots^2}})$$

  mit S_3 Zweien. (Ackermannfunktion)

Analog für Zeit.

Beweis des GAP-Theorems: Sei $M_0, M_1, M_2, \ldots$ eine berechenbare Aufzählung aller TM. Sei $S_i(n) = \max_{|x|=n} space_{M_i}(x)$ (kann unendlich sein).

Lemma: Die Menge $\{(i, n, m) \mid S_i(n) = m\}$ ist entscheidbar.

Beweis des Lemmas: Für jedes Tupel (i, n, m) gibt es eine berechenbare Zeitschranke $t \in \mathbb{N}$, so dass M_i auf Einaben der Länge n nicht mehr als t Schritte machen kann, ohne in eine unendliche Schleife zu geraten und ohne mehr als m Speicherzellen zu gebrauchen. (siehe auch Satz 3.6). Durch Simulation von t Schritten von M_i auf allen Eingaben der Länge n kann so entschieden werden, ob (i, n, m) in der betracheten Menge liegt. $\square_{\text{Lemma}}$

Sei $P = \{(i, n, y) \in \mathbb{N}^3 \mid y < S_i(n) \leq g(y)\}$. P ist entscheidbar. (Lemma 3.13 und Berechenbarkeit von g).

S wird durch den folgenden Algorithmus definiert:

```
Input: n
y := 1
while (∃i ≤ n) (i, n, y) ∈ P do
   waehle kleinstes i ≤ n mit (i, n, y) ∈ P
   y := S_i(n)
od
S(n) := y
```

Es gilt: Aus $S_i(n) \leq g(S(n))$ und $i \leq n$, folgt $S_i(n) \leq S(n)$. $(*)$

Da P entscheidbar ist, ist S berechenbar.

z.Z.: $\mathsf{DSPACE}(g \circ S) \subseteq \mathsf{DSPACE}(S)$.

Sei $L \in \mathsf{DSPACE}(g \circ S)$. Dann ist $L = E(M_i)$ für ein i mit $S_i(n) \leq g(S(n))$ für alle $n \in \mathbb{N}$. Mit $(*)$ folgt für alle $n \geq i$, dass $S_i(n) \leq S(n)$.

Also, $S_i(n) \leq S(n)$ für fast alle n. Es folgt, dass $L = E(M_i) \in \mathsf{DSPACE}(S)$. $\square$

Völlig analog:

Satz: Für jede totale und berechenbare Funktion $g\colon \mathbb{N} \to \mathbb{N}$ mit $g(n) \geq n$ gibt es eine totale und berechenbare Funktion $T\colon \mathbb{N} \to \mathbb{N}$, so dass

$$\mathsf{DTIME}(T) = \mathsf{DTIME}(g \circ T).$$

Man möchte diese Patologien vermeiden!

3.3 Hierarchiesätze

Wir suchen also Klassen von Funktionen T bzw. S, zu denen es kein g gibt mit den obigen Eigenschaften.

Sei $\mathcal{M}$ eine Klasse von abstrakten Maschinen (z.B. alle 2-Band-TM) und R sei eine für $\mathcal{M}$ definierte Resource (z.B. Zeit oder Platz), so dass für jedes $M \in \mathcal{M}$ und jede Eingabe x für M

$$R_M(x) \in \mathbb{N} \cup \{\infty\}$$

Für $T\colon \mathbb{N} \to \mathbb{N}$ ist $R(T)$ eine Komplexitätsklasse:

$$R(T) = \{L \mid \text{ex. TM } M \in \mathcal{M} \text{ mit } R_M(x) \leq T(|x|) \text{ und } L = E(M)\}.$$

Kodierung σ, welche Maschinen aus $\mathcal{M}$ durch Wort $\sigma(M) \in \{0,1\}^*$ kodiert, so dass das Berechnungsverhalten von M aus $\sigma(M)$ effektiv erschlossen werden kann.

$T, t\colon \mathbb{N} \to \mathbb{N}$, $\mathcal{M}_1$, $\mathcal{M}_2$ Klassen von Akzeptoren, R, r Resourcen, die für M_1 bzw. M_2 definiert sind. Das heisst wir haben Koplexitätsklassen $R(T)$ und $r(t)$.

Definition: $R(T)$ *erlaubt Diagonalisierung* über $r(t)$, wenn eine Maschine $D \in \mathcal{M}_1$ existiert mit

- D ist in Resoucen R T-beschränkt und hält auf allen Eingaben. Also: $E(D) \in R(T)$.
- Für jede Maschine $M \in \mathcal{M}_2$, welche in Ressourcen r t-beschränkt ist (d.h. $E(M) \in r(t)$) gilt für fast alle $x \in \{0,1\}^*$

$$\sigma(M)\#x \in E(D) \Leftrightarrow \sigma(M)\#x \notin E(M).$$

Allgemeiner Hierarchiesatz: Wenn $R(T)$ Diagonalisierungen über $r(t)$ erlaubt, dann ist $R(T) \setminus r(t) \neq \emptyset$.

Beweis: Sei D die Diagonalisierungsmaschine aus obiger Definition. Z.Z. $E(D) \notin r(t)$. Sonst gäbe es $M \in \mathcal{M}_2$ mit $E(M) = E(D) \in r(t)$. Das ist jedoch ein Widerspruch dazu, dass $\sigma(M)\#x \in E(D) \Leftrightarrow \sigma(M)\#x \notin E(M)$ für fast alle x. □

Definition: Eine Funktion $T\colon \mathbb{N} \to \mathbb{N}$ heißt *(voll-)zeitkonstruierbar*, falls eine TM M ex., so dass $time_M(x) = T(|x|)$ für alle Eingaben x. ("M kann als Uhr verwendet werden")

Analog:

Definition: Eine Funktion $S\colon \mathbb{N} \to \mathbb{N}$ heißt *(voll-)platzkonstruierbar*, falls eine TM M ex., so dass $space_M(x) = S(|x|)$ für alle Eingaben x.

Ähnliche Definitionen in der Literatur:

Definition von Papadimitriov: Zeit- und Platzkonstruierbarkeit werden zusammen gefasst:
Die Funktion $T\colon \mathbb{N} \to \mathbb{N}$ ist *"gut"* (engl. *"proper"*), wenn $1^n \mapsto 1^{T(n)}$ berechenbar ist in $\mathcal{O}(n + T(n))$ und Platz $O(T(n))$.

Satz: Seien $T, t\colon \mathbb{N} \to \mathbb{N}$, so dass $T(n) \geq n$, T zeitkonstruierbar und $t = o(T)$ $[\frac{t}{T} \longrightarrow_{n\to\infty} 0]$. Dann ist für alle $k \in \mathbb{N}$

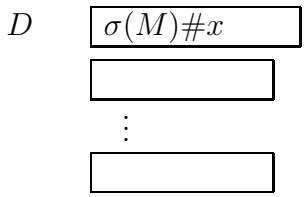
$$\mathsf{DTIME}_k(t) \subsetneq \mathsf{DTIME}(T).$$

Beweis: Wir zeigen: $\mathsf{DTIME}(T)$ erlaubt Diagonalisierungen über $\mathsf{DTIME}_k(t)$.

Zur Erinnerung von Diagonalisierung: Wir suchen D, so dass $\sigma(M)\#x \in E(D) \Leftrightarrow \sigma(M)\#x \notin E(M)$.

Dazu: D sei eine TM mit folgenden Eigenschaften:

a) Auf Eingabe $\sigma(M)\#x$ simuliert D Berechnung von M auf $\sigma(M)\#x$.



b) Für jede k-Band TM M gibt es eine Konstante c_M, so dass D für die Simulation eines Schrittes von M höchstens c_M Schritte benötigt. [etwa $c \cdot |\sigma(M)|$ für ein c].

c) Gleichzeitig simuliert D auf seperaten Bändern eine TM N, welche auf Eingaben der Länge n genau $T(n)$ Schritte macht. Dies ist möglich, da T zeitkonstruierbar ist. "N ist eine Uhr."

d) Nach $T(n)$ Schritten ($n = |\sigma(M)\#x|$) hält D und akzeptiert genau dann, wenn die Berechnung von M auf $\sigma(M)\#x$ bereits verworfen hat.

Es gilt:

- $E(D) \in \mathsf{DTIME}(T)$ ("Uhr")
- für alle M gilt: $T(n) \geq c_M t(n)$ für fast alle n, da $\frac{t(n)}{T(n)} \to 0$.

Also kann D, für fast alle $\sigma(M)x$ die Berechnung von M auf $\sigma(M)\#x$ vollständig in $T(n)$ Schritten simulieren und daher gilt für alle x:

$$\sigma(M)\#x \in E(D) \Leftrightarrow \sigma(M)\#x \notin E(M).$$

□

Korollar (Zeithierarchiesatz): Falls $T(n) \geq n$, T zeitkonstruierbar und $t^2 = o(T)$, dann ist

$$\mathsf{DTIME}(T) \setminus \mathsf{DTIME}(t) \neq \emptyset.$$

Beweis: Nach vorherigem Korollar gibt es eine Konstante c, so dass

$$\mathsf{DTIME}(t) \subseteq \mathsf{DTIME}_1(c \cdot t^2).$$

Wenn $t^2 = o(T)$, dass auch $c \cdot t^2 = o(T)$. Nach vorherigem Satz gibt es ein $L \in \mathsf{DTIME}(T) \setminus \mathsf{DTIME}_1(c \cdot t^2)$. Es folgt, dass

$$L \in \mathsf{DTIME}(T) \setminus \mathsf{DTIME}(t).$$

□

Anwendungen

$$\mathsf{DTIME}(2^n) \subsetneqq \mathsf{DTIME}(n \cdot 2^{2n}),$$

da $(2^n)^2 = o(n \cdot 2^{2n})$.

Für jede monoton wachsende Funktion f mit $\lim_{n\to\infty} f(n) = \infty$ gilt

$$\mathsf{DTIME}(n^{\mathcal{O}(1)}) \subsetneqq \mathsf{DTIME}(n^{f(n)}).$$

z.B. für $f(n) = \log n$ oder $\log\log n$ oder $\log\log\log n$ oder $f(n) = \log^*(n) = \min\{i \mid n \leq \underbrace{n^{2^{\cdot^{\cdot^{\cdot^2}}}}}_{i \text{ mal hoch } 2}\}$.

Scharfe Form des Zeithierarchiesatzes folgt, wenn anstelle des benutzten Korollars eine Simulation von k-Band-TM durch 2-Band-TM verwendet wird.

Satz: Für $T(n) \geq n$ gilt:

$$\mathsf{DTIME}(T) \subseteq \mathsf{DTIME}_2(T \cdot \log T).$$

Beweis: Siehe Übung.

Korollar (Zeithierarchiesatz – Scharfe Form): Falls $T(n) \geq n$, T zeitkonstruierbar und $t \cdot \log t = o(T)$, dann ist

$$\mathsf{DTIME}(T) \setminus \mathsf{DTIME}(t) \neq \emptyset.$$

Satz (Platzhierarchiesatz): Seien $S, s \colon \mathbb{N} \to \mathbb{N}$ mit $S(n) \geq \log n$, S platzkonstruierbar und $s = o(S)$, dann ist

$$\mathsf{DSPACE}(S) \setminus \mathsf{DSPACE}(s) \neq \emptyset.$$

Beweis: Da wir die Anzahl der Arbeitsbänder ohne Platzverlust auf eins reduzieren können, reicht es TM mit einem Eingabeband und einem Arbeitsband zu betrachten.

Sei M s-platzbeschränkt. Also gibt es höchstens $|Q| \cdot (n+1) \cdot |\Sigma|^{s(n)} \cdot s(n) = (n+1) \cdot 2^{\mathcal{O}(s(n))}$ partielle Konfigurationen auf Eingaben der Länge n. Also hält M nach $\leq t(n) = 2^{c_M \cdot s(n) + \log n}$ Schritten. Da S platzkonstruierbar ist, existiert eine TM N mit $\mathrm{space}_N(x) = S(|x|)$ für alle x.

Wir wollen zeigen, dass $\mathsf{DSPACE}(S)$ Diagonalisierung über $\mathsf{DSPACE}(s)$ erlaubt.

D operiert auf Eingabe $\sigma(M)\#x$ wie folgt:

a) Zuerst wird durch Simulation von N ein Bereich $S(n)$ Speicherzellen auf dem Arbeitsband markiert. Die folgenden Operationen werden innerhalb dieses Bereiches durchgeführt. Bei Überschreitung des Bereiches hält D und verwirft.

b) D initialisiert einen Zähler auf $t(n)$ und wird auf spezielle Spur des Bandes geschrieben.

c) D simuliert Berechnung von M auf $\sigma(M)\#x$ und vermindert bei jedem simulierten Schritt, den Zähler um Eins.

d) Falls die Simulation mehr als die markierten Speicherzellen benötigt, oder m nicht in $t(n)$ Schritten hält, dann verwirft D die Eingabe $\sigma(M)\#x$. Wenn D erfolgreich eine *verwerfende* Berechnung von M auf $\sigma(M)\#x$ simuliert hat, dann aktzeptiert D die Eingabe $\sigma(M)\#x$.

- $E(D) \in \mathsf{DSPACE}(S)$.
- Behauptung: Wenn M s-platzbeschränkt und auf jeder Eingabe hält, dann kann D für fast alle x die Berechnung von M auf $\sigma(M)\#x$ vollständig simulieren.

  Dazu:

  - Da $t(n) \leq 2^{c_n \cdot \log(s(n)) + \log n} \leq 2^{2 \cdot S(n)} = 2^{S(n)}$ für hinreichend große n, da $\frac{s(n)}{S(n)} \to 0$, kann der Zähler durch Wörter der Länge $\leq S(n)$ implementiert werden.
  - M habe ein Alphabet mit d_M Symbolen. D braucht $\leq \log_{|\Sigma_D|} d_n = l_M$ Felder, um ein Symbol von M zu kodieren. Also braucht D zur Simulation von M den Platz $\leq l_M \cdot \text{space}_M(\sigma(M)\#x) \leq l_M \cdot s(n) \leq S(n)$ für hinreichend großes n.

  Nun sieht man leicht, dass

  $$\sigma(M)\#x \in E(D) \Leftrightarrow \sigma(M)\#x \notin E(M)$$

  für fast alle x. Also erlaubt $\mathsf{DSPACE}(S)$ Diagonalisierbarkeit über $\mathsf{DSPACE}(s)$. □

Folgerung: $\mathsf{LOGSPACE} = \mathsf{DSPACE}(\mathcal{O}(\log n))$ $\mathsf{PSPACE} = \bigcup_{c \in \mathbb{N}} \mathsf{DSPACE}(n^c)$ Aus dem Platzhierarchiersatz folgt:

$$\mathsf{LOGSPACE} \subsetneq \mathsf{PSPACE}$$

$$\mathsf{LOGSPACE} \subseteq \mathsf{P} \subseteq \mathsf{PSPACE}$$

3.4 GAP-Theoreme für kleine Komplexitätsklassen

$\mathsf{REG} := \{L \mid L \text{ regulär}\}$.

Satz: $\mathsf{REG} = \mathsf{DSPACE}(0) = \mathsf{DSPACE}(\mathcal{O}(1))$.

Beweis: Zu $\mathsf{REG} \supseteq \mathsf{DSPACE}(0)$: Zweiwege-Automaten können durch Einwege-Automaten simuliert werden. □

Satz: Sei $s(n) = o(\log \log n)$. Dann ist $\mathsf{DSPACE}(s) = \mathsf{DSPACE}(\mathcal{O}(1))$.

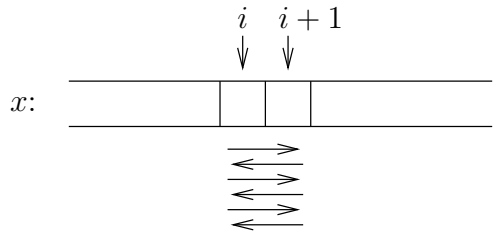
Beweis:

„$\supseteq$“ trivial.

„$\subseteq$“ Sei M TM mit Eingabeband und einem Arbeitsband (Œ, da $\mathsf{DSPACE}(s) = \mathsf{DSPACE}_1(s)$), die Zustandsmenge sei Q, das Alphabet Σ, welche auf jede Eingabe hält und $s(n) = \max_{|x|=n} \mathrm{space}_M(x) \overset{\text{Vor.}}{=} o(\log \log n)$. Zu zeigen ist dann, dass $s(n) = \mathcal{O}(1)$.

Erweiterte Zustände (EZ) sind Tupel (q, p, w). Die Anzahl der erweiterten Zustände ist $\tilde{s}(n) := |Q| \cdot s(n) \cdot |\Sigma|^{s(n)}$.

Eingabeband:



Sei $\mathrm{ECS}(x, i)$ die erweiterte Crossing-Sequenz von M auf x und der Stelle i. Dies ist die Folge der erweiterten Zustände von M bei Übergängen zwischen i-ten und $(i+1)$-ten Eingabesymbol. In keiner ECS kann ein EZ mehr als zweimal auftreten (einmal in jeder Richtung), sonst gäbe es eine unendliche Schleife. Also

$$|\mathrm{ECS}(x, i)| \leq 2 \cdot \tilde{s}(n).$$

($|\mathrm{ECS}(x, i)|$ bezeichnet die Länge der Folge). Also gibt es höchstens

$$f(n) := \sum_{m=0}^{2\tilde{s}(n)} \tilde{s}(n)^m < 2 \cdot \tilde{s}(n)^{2 \cdot \tilde{s}(n)+1}$$

verschiedene ECS für jedes x und i.

Annahme: Für alle k existiert ein n, so dass $s(n) > k$. Wir werden ein Pumping-Argument benutzen, um einen Widerspruch herzuleiten. Es gibt eine Konstante c, so dass

$$\tilde{s}(n) \leq 2^{c \cdot s(n)} \text{ und } f(n) \leq 2^{2^{c \cdot s(n)}}$$

für hinreichend großes n gilt. Da $s(n) = o(\log\log n)$ gibt es ein n_0, so dass für alle $n \geq n_0$

$$s(n) \leq \frac{\log\log n}{2c}. \qquad (*)$$

Wähle nun eine Eingabe x von minimaler Länge $n_1 > n_0$ derart, dass

$$\text{space}_M(x) = s(n_1) > \max_{n \leq \max(n_0,16)} s(n).$$

Nach der Annahme existiert ein solches x. Die Anzahl der ECS von x ist höchstens

$$f(n_1) \leq 2^{2^{c \cdot s(n_1)}} \overset{(*)}{\leq} 2^{2^{\frac{1}{2}\log\log n_1}} \overset{\forall n>4: \frac{1}{2}\log n < \log\frac{n}{2}}{<} 2^{2^{\log\log\frac{n_1}{2}}} = \frac{n_1}{2}.$$

Es gibt also mindestens drei Positionen $a, b, c < n_1$ mit $a < b < c$, so dass

$$\text{ECS}(x, a) = \text{ECS}(x, b) = \text{ECS}(x, c).$$

x ist die kürzeste Eingabe mit Konfiguration C und Platzverbrauch $s := s(n_1)$. i sei die Position des Eingabekopfes bei der Konfiguration C.

Fall 1: $i \leq b$. Betrachte $\hat{x} = x_0 \ldots x_b x_{c+1} \ldots x_{n-1}$. Da $\text{ECS}(x, b) = \text{ECS}(x, c)$ sieht Berechnung von M auf $\hat{x}$ gleich aus, wie Berechnung von M auf x, ausser, dass alle Konfigurationen, in denen der Lesekopf im Segment $x_{b+1} \ldots x_c$ steht, wegfallen.

Konfiguration C ist immer noch da.

Fall 2: $i > b$. Analog. □

Bemerkung: Für $s(n) = o(\log\log n)$ folgt aus den beiden obigen Sätzen, dass

$$\mathsf{DSPACE}(s) = \mathsf{REG}.$$

Satz: Sei $t = o(n \log n)$. Dann gilt,

$$\mathsf{DTIME}_1(t) \subseteq \mathsf{DTIME}_1(\mathcal{O}(n)).$$

(1-Band heisst hier einschließlich Eingabeband).

Beweis: M sei $o(n \log n)$, aber nicht $\mathcal{O}(n)$-zeitbeschränkte 1-Band TM, welche ohne Einschränkung bei jedem Schritt den Kopf bewegt. Eine *Crossing-Sequenz* $\text{CS}(x, j)$ ist die Folge der Zustände, in denen M auf Eingabe x die Grenzen zwischen den Feldern j und $j + 1$ überschreitet.

1. M erzeugt C beliebiger Länge.
   Denn: Sonst gilt $|\mathrm{CS}(x,j)| \leq m$ für alle x und j für ein m.

   Dann kann M im Eingabebereich höchstens $n \cdot m$ Schritte durchführen. Für alle $i > n$ und $j > 0$ muss gelten, dass $\mathrm{CS}(x,i) \neq \mathrm{CS}(x,i+j)$, sonst rechnet M unendlich lange, da sich die Berechnung auf Feldern i bis $i+j$, $i+j$ bis $i+2j$, ... wiederholt. Also ist M $n+\mathcal{O}(1)$-platzbeschränkt und $n \cdot m + \mathcal{O}(1)$ $(= \mathcal{O}(n))$ zeitbeschränkt. Widerspruch zur Annahme, dass M nicht $\mathcal{O}(n)$-zeitbeschränkt ist.

   Für unendlich viele m existiert also $x(m)$ von minimaler Länge $n(m)$, so dass M auf Eingabe $x(m)$ CS der Länge m erzeugt. Jede CS auf $x(m)$ in den Positionen $0, \ldots, x(m)-1$ kann höchstens zweimal auftreten. Sonst könnten wir ein Segment aus $x(m)$ rausschneiden und $x(m)$ verkürzen. Also erzeugt M auf $x(m)$ mindestens $k = \left\lfloor \frac{n(m)}{2} \right\rfloor$ verschiedene CS $\sigma_1, \ldots, \sigma_k$ und es gilt

$$\mathrm{time}_M(x(m)) \geq \sum_{i=1}^{k} |\sigma_i|.$$

   Zu zeigen ist, dass es eine Konstante $c > 0$ gibt, so dass

$$\sum_{i=1}^{k} |\sigma_i| \geq c \cdot n(m) \cdot \log\log n(m). \qquad (*)$$

   Dies wäre ein Widerspruch zur Annahme, dass M $o(n \log n)$-zeitbeschränkt ist.

   Beweis von $(*)$: Für $l \in \mathbb{N}$ gibt es $\leq |Q|^l$ verschiedene CS der Länge l.

$$\sum_{i=1}^{k} |\sigma_i| = \sum_{l=0}^{\max_i(|\sigma_i|)} |\{j \mid |\sigma_j| = l\}| \cdot l$$

   ist minimal, wenn es möglichst viele CS kleiner Länge gibt. Wenn r die Bedingung $\sum_{l=0}^{r} |Q|^l \leq k$ erfüllt, gilt also

$$\sum_{i=1}^{k} |\sigma_i| \geq \sum_{l=0}^{r} |Q|^l \cdot l.$$

Setze $r := (\log_{|Q|} k) - 1$. Dann gilt:

$$\sum_{l=0}^{r} |Q|^l = \frac{|Q|^{r+1} - 1}{|Q| - 1} = \frac{|Q|^{\log_{|Q|} k} - 1}{|Q| - 1} \leq \frac{k-1}{|Q|-1} \leq k.$$

Also

$$\begin{aligned}\sum_{i=1}^{k} |\sigma_i| &\geq \sum_{l=0}^{r} |Q|^l \cdot l \geq |Q|^r \cdot r = \frac{|Q|^{\log_{|Q|} k}}{|Q|}((\log_{|Q|} k) - 1) \\ &= \frac{k}{|Q|}(\log_{|Q|} k - 1) \overset{k=\frac{n(m)}{2}}{\geq} c \cdot n(m) \cdot \log n(m)\end{aligned}$$

für ein geeignetes c. □

4 Nichtdeterministische Komplexitätsklassen

4.1 Nichtdeterministische Turingmaschine

Definition: *Nichtdeterministische Turingmaschinen (NTM)* sind analog zu Turingmaschinen (TM) definiert, außer, dass die Übergangsfunktion im Allgemeinen mehrere Übergänge zulässt. Das wichtigste Modell ist ein k-Band Akzeptor. Seine Übergangsfunktion hat die Form

$$\delta \colon (Q \setminus F) \times \Sigma^k \to 2^{Q \times \Sigma^k \times \{-1,0,1\}^k}.$$

Sei C eine *Konfiguration* (definiert wie üblich) von M. Dann ist

$$Next(C) := \{C' \mid C \vdash C'\}$$

die *Menge der Nachfolgekonfigurationen* von C.
Eine *Berechnung* ist eine Folge von Konfigurationen $C_0, \ldots, C_t, \ldots$ mit $C_{i+1} \in Next(C_i)$.
Mit $T_{M,x}$ wird der *Berechnungsbaum* von M auf x bezeichnet.

Bemerkung: Für jede NTM M gibt es ein $r \in \mathbb{N}$ mit $|Next(C)| \leq r$ für alle C. (Wegen $Next(C) \subseteq 2^{Q \times \Sigma^k \times \{-1,0,1\}^k}$)

Definition: Eine NTM heißt *T-zeitbeschränkt* (bzw. *S-platzbeschränkt*), wenn für alle Eingaben x gilt, dass keine Berechnung von M auf x mehr als $T(|x|)$ Schritte macht (bzw. mehr als $S(|x|)$ Speicherplatz braucht).

Definition: Sei M eine NTM und x eine Eingabe für M. M *akzeptiert* x, wenn es eine Berechnung von M auf x gibt, welche in einer akzeptierenden Konfiguration endet.
$E(M) := \{x \mid M \text{ akzeptiert } x\}$.

Definitionen:
- $\mathsf{NTIME}(T) := \{L \mid \text{es ex. } T\text{-zeitbeschr. NTM } M \text{ mit } E(M) = L\}$.
- $\mathsf{NSPACE}(S) := \{L \mid \text{es ex. } S\text{-platzbeschr. NTM } M \text{ mit } E(M) = L\}$.

4.2 Elementare Eigenschaften von nicht-det. Komplexitätsklassen

Satz: Für alle $T\colon \mathbb{N} \to \mathbb{R}^+$ gilt

$$\mathsf{DTIME}(T) \subseteq \mathsf{NTIME}(T) \subseteq \mathsf{DTIME}(2^{\mathcal{O}(T)}).$$

Satz von Savitch: Sei $S(n) \geq \log n$ platzkonstruierbar. Dann ist

$$\mathsf{NSPACE}(S) \subseteq \mathsf{DSPACE}(S^2).$$

Folgerung:

$$\bigcup_{c\in\mathbb{N}} \mathsf{NSPACE}(n^c) =: \mathsf{NPSPACE} = \mathsf{DPSPACE} := \bigcup_{c\in\mathbb{N}} \mathsf{DSPACE}(n^c).$$

4.3 Der Satz von Immerman und Szelepcsényi

Definition: Sei $\mathcal{C}$ eine Klasse von Sprachen. Dann bezeichnen wir mit co-$\mathcal{C} := \{\bar{L} \mid L \in \mathcal{C}\}$ die komplementäre Klasse. $\bar{L}$ ist das Komplement von L.

Bemerkung: Deterministische Komplexitätsklassen ($\mathsf{DTIME}(T)$, $\mathsf{DSPACE}(S)$) sind unter Vereinigung, Durchschnitt und Komplement abgeschlossen.

$\mathsf{NTIME}(T)$, $\mathsf{NSPACE}(S)$ sind abgeschlossen unter Durchschnitt und Vereinigung.

$\mathsf{NTIME}(T)$ ist vermutlich nicht unter Komplement abgeschlossen. Insbesondere gilt dann $\mathsf{NP} \neq \mathsf{co\text{-}NP}$.

Satz (Immerman, Szelepcsényi 1987): Sei $S(n) \geq \log n$ und platzkonstruierbar. Dann gilt

$$\mathsf{NSPACE}(S) = \mathsf{co\text{-}NSPACE}(S).$$

Definition: Ein *irrtumsfreies* nicht-deterministisches Berechnungsverfahren für eine Funktion f ist eine nicht-deterministische Turingmaschine M, mit

(i) Jede Berechnung von M auf x hält und produziert als Ausgabe entweder $f(x)$ oder ? (steht für "weiß nicht").
(ii) Mindestens eine Berechnung von M auf x ergibt $f(x)$.

Definition: Ein *irrtumsfreies* Entscheidungsverfahren für eine Sprache $L \subseteq \Gamma^*$ ist ein irrtumsfreies Berechnungsverfahren für χ_L (charakteristische Funktion von L).

Satz: Sei $S(n) \geq \log n$ platzkonstruierbar. Dann gibt es für jedes $L \in \mathsf{NSPACE}(S)$ auch ein irrtumsfreies Entscheidungsverfahren, das S-platzbeschränkt ist.

Bemerkung: Daraus folgt direkt der Satz von Immerman und Szelepcsényi. Man vertauscht einfach die Ausgaben "ja" und "nein". Bei ? wird immer verworfen.

Beweis: M sei $S(n)$-platzbeschränkte nicht-deterministische TM. Sei $L = E(M)$. $C_0(x)$ sei die Anfangskonfiguration von M auf Eingabe x. $\mathcal{C}_{S(n)}$ sei die Menge der Konfigurationen von M mit Platzverbrauch $\leq S(n)$. Da $S(n) \geq \log n$: Jede Konfiguration $C \in \mathcal{C}_{S(n)}$ durch Wert der Länge $c \cdot S(n)$ für ein c beschreibbar ist.

Sei $Reach_x(t) := \{C \in \mathcal{C}_{S(n)} \mid C \text{ ist von } C_0(x) \text{ in} \leq t \text{ Schritten erreichbar}\}$.

$R_x(t) := |Reach_x(t)| \leq 2^{\mathcal{O}(S(n))}$.

1. Es gibt nicht-det. Algorithmus M_0 mit Eingabe x, r, t, C, wobei x Eingabe für M, $r, t \in \mathbb{N}$, $C \in \mathcal{C}_{S(n)}$ ($n = |x|$), so dass gilt: Wenn $r = R_x(t)$, dann entscheidet M_0 irrtumsfrei mit Platz $\mathcal{O}(S(n))$, ob $C \in Reach_x(t+1)$.

Algorithmus $M_0(x, r, t, C)$:

```
m := 0;
for all D ∈ C_{S(n)} do {
  Simuliere hoechstens t Schritte von M auf x
    (nicht det.)
  if (D wurde erreicht) {
    m := m+1;
    if (C ∈ Next(D))
      return 1; }}
if (m = r) return 0;
else return ?;
```

Korrektheit: Sei $r = R_x(t)$.

Wenn $C \in Reach_x(t+1)$ folgt, dass es eine Berechnung von M_0 mit Ausgabe 1 gibt und keine Berechnung liefert 0.

Umgekehrt, wenn $C \notin Reach_x(t+1)$ folgt, dass es eine Berechnung von M_0 mit Ausgabe 0 gibt und keine Berechnung liefert 1.

2. Es gibt eine Funktion $t(n) = 2^{\mathcal{O}(S(n))}$, so dass M x entweder in $\leq t(|x|)$ Schritten akzeptiert oder verwirft.

Lemma: Es gibt ein irrtumsfreies nicht-det. $\mathcal{O}(S(n))$-platzbeschränktes Berechnungsverfahren für die Funktion $x \mapsto R_x(t(|x|) - 1)$.

Beweis: durch induktives Zählen.

Algorithmus $M_1(x)$:

```
r := 1;
for t = 0 to t(|x|) − 2 do {
  m := 0;
  for all C ∈ C_{S(n)} do {
    z := M_0(x, r, t, C);
    if (z = 1) m := m+1;
    if (z = ?) return ?; }
  r := m; }
return r;
```

Korrektheit: Immer, wenn $M_0(x, r, t, C)$ aufgerufen wird, gilt $r = R_x(t)$. Für $t = 0$ ist $r = 1 = R_x(0)$. Für $t > 0$ ist $r = |\{C \mid$

es gibt Berechnung von M_0 auf Eingabe $(x, R_x(t-1), t-1, C)$ mit Ausgabe $1\}|$.
Insbesondere gilt, dass der Wert von r am Ende einer erfolgreichen Berechnung von M_1 ist $R_x(t(|x|-1)$. □

3. Irrtumsfreies nicht-deterministisches Entscheidungsverfahren für $L = E(M)$:

   Algorithmus $\tilde{M}(x)$:

```
r := M_1(x);
if (r = ?) return ?;
for all (C_a ∈ C_{S(n)}, C_a akzeptiert) {
  z := M_0(x, r, t(|x|) − 1, C_a);
  if (z = 1) return "x ∈ L";
  if (z = ?) return ?; }
return "x ∉ L";
```

   Sei $x \in L$. Dann gibt es eine Berechnung von M_1 mit $r = R_x(t(|x|)-1)$ und es gibt eine akzeptierende Konfiguration C_a mit $C_a \in Reach_x(t(|x|))$ und also eine Berechnung von M_0 auf $(x, r, t(|x|) - 1, C_a$ mit Ausgabe 1. Es folgt, dass es eine Berechnung von $\tilde{M}$ auf x gibt, die die Antwort "$x \in L$" liefert.
   Es wird niemals $x \notin L$ ausgegeben, da es eine erreichbare akzeptierende Konfiguration C_a gibt und für diese liefert $M_0(x, r, t(|x|) - 1, C_a)$ höchstens 1 oder ?, aber nicht 0.

   Sei $x \notin L$. Es gibt Berechnung von $\tilde{M}$ mit Ausgabe "$x \notin L$", denn es gibt eine Berechnung von M_1 mit Ausgabe $R_x(t(|x|) - 1)$ und für alle C_a gibt es eine Berechnung von M_0 mit Ausgabe 0.
   Keine Berechnung liefert "$x \in L$", weil kein C_a erreichbar ist und M_0 wird mit $r = R_x(|x| - 1)$ aufgerufen und ist irrtumsfrei, liefert also niemals 1.

Es folgt, dass $\tilde{M}$ ein irrtumsfreies Entscheidungsverfahren für L ist. Man sieht leicht, dass $\tilde{M}$ $\mathcal{O}(S(n))$-platzbeschränkt ist.

Mit dem Bandkompressions Satz folgt, dass $\tilde{M}$ in Platz $S(n)$ implementiert werden kann. □

Folgerung: CSL (kontext-sensitve Sprachen) = NSPACE(n) ist unter Komplement abgeschlossen.

Folgerungen:
- NLOGSPACE = co-NLOGSPACE.

- NSPACE = co-NSPACE.

5 NP-Probleme

5.1 Die Klassen P und NP

Definition: $\mathsf{P} = \mathsf{PTIME} = \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}(n^k)$.

1. P ist die Klasse der "effizient" oder "praktisch" lösbaren Problemen.
2. P ist robust. Fast alle vernünftigen Maschinenmodelle definieren die selbe Klasse.
3. P hat angenehme Abschlusseigenschaften: Vereinigung, Schnitt, Komplement, transitive Hülle (A^*), polynomielle Reduktion.

Definition: Sei $A \subseteq \Sigma^*$, $B \subseteq \Gamma^*$. Wir sagen A kann auf B *reduziert* werden (in Zeichen $A \leq^p B$), falls es eine polynomiell berechenbare Funktion $f\colon \Sigma^* \to \Gamma^*$ existiert, so dass für jedes $x \in \Sigma^*$ gilt

$$x \in A \text{ genau dann, wenn } f(x) \in B.$$

Es gilt

- Falls $A \leq^p B$ und $B \in \mathsf{P}$, dann ist auch $A \in \mathsf{P}$.
- Die Relation $\leq^p$ ist transitiv, d.h. falls $A \leq^p B$ und $B \leq^p C$, dann gilt auch $A \leq^p C$.

Viele wichtige Probleme liegen nicht in P. So zum Beispiel, dass Traveling-Salesman-Problem:

Traveling-Salesman-Prolem (TSP):

$$\mathrm{TSP}(D) := \{(D,k) \mid D \in \mathbb{N}^{n\times n}, k \in \mathbb{N}, \exists \text{ zyklische Permutation } \pi \text{ über } \{1,\ldots,n\} \text{ mit } \sum_{i=1}^{n} d_{i,\pi(i)} \leq k\}.$$

Es ist offen, ob $\mathrm{TSP}(D)$ in Pist.

Aber es ist leicht zu verifizieren, ob eine gegebene zyklische Permutation die Bedingung $\sum_{i=1}^{n} d_{i,\pi(i)} \leq k$ erfüllt.

$$\text{TOUR} := \{(D,k,\pi) \mid D,k \text{ wie bei TSP, } \pi \text{ zyklische Permutation } \sum_{i=1}^{n} d_{i,\pi(i)} \leq k\}.$$

Wenn $(D,k,\pi) \in \text{TOUR}$, dann nennt man π einen *polynomiellen Zeugen* dafür, dass $(D,k) \in \text{TSP}(D)$.

Ein anderes Beispiel ist SAT. Hier ist eine erfüllende Belegung ein polynomieller Zeuge.

Definition: Sei $L \subseteq \Sigma^*$. $L \in \mathsf{NP}$ genau dann, wenn es $L_0 \in \mathsf{P}$ und ein Polynom $p(n)$ gibt, so dass $L = \{x \mid \exists y \text{ mit } |y| \leq p(|x|) \text{ und } x\#y \in L_0\}$.
y ist dann ein *polynomieller Zeuge* für $x \in L$.

Notation: $\exists^p y$ für $\exists y$ mit $|y| \leq p(|x|)$ und $x\#y \in L_0$.

Es gilt $\text{TSP}(D)$, $\text{SAT} \in \mathsf{NP}$.

Satz: Es gilt
$$\mathsf{NP} = \bigcup_{k \in \mathbb{N}} \mathsf{NTIME}(n^k).$$

Beweis:

„$\subseteq$“ Sei $L = \{x \mid \exists^p y(x\#y) \in L_0\} \in \mathsf{NP}$ mit Polynom $p(n)$, $L_0 \in \mathsf{P}$. Sei M_0 q-zeitbeschränkte TM, $q(n)$ ein Polynom mit $L_0 = E(M_0)$. Wir geben jetzt eine NTM M für L an:

Eingabe: x.
Rate nicht deterministisch y mit $|y| \leq p(|x|)$.
Gebe $M_0(x\#y)$ aus.

Offensichtlich gilt $E(M) = L$. Ausserdem ist M $p(|x|) + q(|x| + 1 + p(|x|))$-zeitbeschränkt. Also gibt es ein Polynom p', so dass M $p'(|x|)$-zeitbeschränkt ist.

„$\supseteq$“ Sei $L = E(M)$, M $p(n)$-zeitbeschränkte 1-Band-NTM, p ein Polynom. Eine *Berechnung* von M auf x ist eine Folge von höchstens $p(|x|)$ Konfigurationen. Jede Konfiguration ist durch ein Wort der Länge $p(|x|)$ kodiert. $n = |x|$.

mtuBild fehlt noch

(q, w, i) mit $w = w_0 \dots w_{p(n)-1}$. Dann ist $w_0 \dots w_{i-1}(q, w_i)w_{i+1} \dots w_{p(n)-1} \in (\Sigma \cup (Q \times \Sigma))^*$.

Also kann die Berechnung durch Wörter der Länge $\mathcal{O}(p^2(n))$ kodiert werden (z.B. Zeilen hintereinander schreiben). Man sieht mit

$$L_0 = \{x\#y \mid y \text{ kodiert akz. Berechnung auf } x\},$$

dass $L_0 \in \mathsf{P}$ ist. Offensichtlich gilt nun:

$$x \in L \Leftrightarrow \exists y\, |y| \leq \mathcal{O}(p^2(n)), x\#y \in L_0.$$

Also ist $L \in \mathsf{NP}$. □

5.2 NP-Vollständigkeit

Satz:
(i) Es gilt $\mathsf{P} \in \mathsf{NP}$.
(ii) Falls $A \leq^p B$ und $B \in \mathsf{NP}$, dann ist auch $A \in \mathsf{NP}$.

Die *Cook'sche Hypothese* behauptet, dass $\mathsf{P} \neq \mathsf{NP}$ ist. Dies gilt als das wichtigste offene Problem der Komplexitätstheorie.

Die "schwierigsten Probleme" in NP liegen nur in P, falls $\mathsf{P} = \mathsf{NP}$. Deshalb sehen wir uns diese Probleme an:

Definition: B ist *NP-hart* (oder *NP-schwer*), wenn für alle $A \in \mathsf{NP}$ gilt, dass $A \leq^p B$.
B ist *NP-vollständig*, wenn $B \in \mathsf{NP}$ und NP-schwer ist.

Satz: Sei B NP-vollständig. Dann ist

$$\mathsf{P} = \mathsf{NP} \text{ genau dann, wenn } B \in \mathsf{P}.$$

Beweis:

„⇒" Klar.

„⇐" Sei A NP-vollständig. Mit obiger Definition und $A \leq^p B$, folgt mit $B \in \mathsf{P}$, dass $A \in \mathsf{P}$ ist. Also wären alle NP-vollständigen Probleme in P. □

Beispiel für ein NP-vollständiges Problem (es gibt sehr sehr viele!):

Grundbegriffe der Aussagenlogik (AL):

Sei V eine abzählbare Menge von Aussagenvariablen. AL ist die Menge der *Formeln* der AL. Sie ist induktiv definiert durch:

- Es gilt $V \subseteq AL$.
- Sind $\psi, \varphi \in AL$, dann sind auch $(\psi \vee \varphi), (\psi \wedge \varphi), \neg\psi \in AL$.

$V(\psi)$-Menge der in ψ auftretenden aussagenlogischen Variablen.

Eine Bewertung ist eine Funktion $\epsilon\colon W \to \{0,1\}$, wobei $W \subseteq V$ ist. 0 steht für false, 1 für true. Man kann eine Bewertung auf aussagenlogische Formeln wie bekannt erweitern.

Wir nennen ψ *erfüllbar*, genau dann, wenn es eine Bewertung $\epsilon\colon V(\psi) \to \{0,1\}$ gibt mit $\epsilon(\psi) = 1$. Die Formel ψ ist eine *Tautologie*, genau dann, wenn für $\neg\psi$ unerfüllbar ist. Dies gilt genau dann, wenn für jede beliebige Bewertung $\epsilon\colon V(\psi) \to \{0,1\}$ gilt, dass $\epsilon(\psi) = 0$.

Wir kodieren Aussagenvariablen X_i durch das Wort X("Binärdarstellung von i"). Jede Formel wird also durch Wörter über dem Alphabet $\Sigma = \{X, 0, 1, (,), \wedge, \vee, \neg\}$ kodiert.

Definition: SAT $= \{\psi \in AL \mid \psi \text{ erfüllbar }\}$.

Es gilt SAT $\in$ NP, denn $\{\psi\#\epsilon \mid \epsilon\colon V(\psi) \to \{0,1\}, \epsilon(\psi) = 1\} \in$ P und ϵ kann linear in der Länge von ψ kodiert werden.

Satz von Cook und Levin: SAT ist NP-vollständig.

Beweis: Wir haben eben eingesehen, dass SAT $\in$ NP ist. Noch zu zeigen bleibt, dass SAT NP-schwer ist. Dazu zeigen wir, dass für alle $A \in$ NP gilt, dass $A \leq^p$ SAT. Sei $M = (Q, \Sigma, q_0, F, \delta)$ mit $F = F^+ \dot\cup F^-$ sei 1-Band-NTM, welche A in Zeit $p(n)$ entscheidet, wobei $p(n)$ ein Polynom ist. O.B.d.A. nehmen wir an, dass jede Berechnung von M in einer akzeptierenden oder verwerfenden Konfiguration endet, d.h. $Next(C) = \emptyset$ genau dann, wenn C eine Endkonfiguration ist. Sei $w = w_0 \dots w_{n-1}$ eine Eingabe für M. Wir versuchen jetzt eine polynomiell berechenbare Funktion f zu finden, so dass für alle $w \in \Sigma^*$ gilt, dass

$$w \in \underbrace{E(M)}_{=A} \iff \underbrace{f(w)}_{=:\psi_w} \in \text{SAT}.$$

ψ_w enthält folgende Aussagenvariablen:

$$X_{q,t} \text{ mit } q \in Q,\ t \leq p(n) \quad (n = |w|).$$

[Intuition: Zur Zeit t befindet sich M im Zustand q.]

$$Y_{a,i,t} \text{ für } a \in \Sigma,\ i, t \leq p(n).$$

[Intuition: Zur Zeit t steht auf dem i-ten Feld der Buchstabe a.]

$$Z_{i,t} \text{ für } i, t \leq p(n).$$

[Intuition: Zur Zeit t steht Schreib-/Lesekopf auf dem i-ten Feld.]

Berechnung von M induziert Belegung der Variablen.

Sei $w = w_0 \dots w_{n-1}$.

$$\psi_w := \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$$

$$\varphi_1 := X_{q_0,0} \wedge \bigwedge_{i=0}^{n-1} Y_{w_i,i,0} \wedge \bigwedge_{i=n}^{p(n)} Y_{\square,i,0} \wedge Z_{0,0}.$$

[Anfangskonfiguration]

$$\varphi_2 := \bigwedge_{t<p(n),\, a\in\Sigma, i\neq j} (Z_{i,t} \wedge Y_{a,j,t} \rightarrow Y_{a,j,t+1}).$$

[Änderung nur an S/L-Kopfposition]

$$\varphi_3 := \bigwedge_{\substack{t<p(n)\\ i,a,q}} \Big((X_{q,t} \wedge Y_{a,i,t} \wedge Z_{i,t}) \rightarrow \bigvee_{\substack{(q',b,m)\in\delta(q,a)\\ 0\leq i+m\leq p(n)}} (X_{q',t+1} \wedge A_{b,i,t+1} \wedge Z_{i+m,t+1}) \Big).$$

[Übergang]

$$\varphi_4 := \bigwedge_{t\leq p(n),\, q\in F^-} \neg X_{q,t}.$$

[niemals wird eine verwerfende Konfiguration erreicht.]

Offensichtlich ist f polynomiell berechnebar.

„⇒“ Sei $w \in E(M)$. Eine akzeptierende Berechnung induziert eine Interpretation $X_{q,t}$, $Y_{a,i,t}$, $Z_{i,t}$, welche ψ_w wahr macht. Also ist ψ_w erfüllbar.

„⇐“ Hier haben wir ein kleines Problem, da in der Formel nicht alles spezifiziert ist. Z.B. könnte es sein, dass $Z_{i,t} = Z_{j,t} = 1$ für $i \neq j$ gilt.

Sei $C = (q, y, l)$ eine Konfiguration von M, $t \leq p(n)$ mit $y = y_0 \dots y_{p(n)}$,

$$\alpha_{C,t} = X_{q,t} \wedge \bigwedge_{i=0}^{p(n)} Y_{y_i,i,t} \wedge Z_{l,t}$$

$[\varphi_1 = \alpha_{C_0(w),0}]$

Also gilt:

1. $\psi_w \vDash \alpha_{C_0(w),0}$ und
2. $\psi_w \wedge \alpha_{C,t} \vDash \bigvee_{C' \in Next(C)} \alpha_{C',t+1}$.

Sei $\epsilon(\psi(w)) = 1$. Aus 1. und 2. folgt, dass mindestens eine Berechnung $C_0(w) = C_0, C_1, \dots, C_r$ $(r \leq p(n))$ existiert mit $\epsilon(\alpha_{C_t,t}) = 1$ für alle $t = 0, \dots, r$. Wegen Teilformel φ_4 von ψ_w gilt für alle verwerfenden Endkonfigurationen C von M und alle $t \leq p(n)$

$$\psi_w \vDash \neg\alpha_{C,t}.$$

Also ist C_r eine akzeptierende Endkonfiguration und damit akzeptiert M das Wort w. □

Bemerkung: Die Reduktion f ist sehr einfach, sie ist insbesondere berechenbar in *logspace*.

Definition: Sei $A \subseteq \Gamma^*$, $B \subseteq \Gamma^*$. Dann ist A *mit logarithmischen Platz reduzierbar* auf B (in Zeichen $A \leq^{\log} B$), wenn eine Funktion $f\colon \Gamma^* \to \Sigma^*$ existiert mit

- $x \in A$ gdw. $f(x) \in B$ für alle $x \in \Gamma^*$ und
- f ist durch eine $\mathcal{O}(\log n)$-platzbeschränkte TM berechenbar.

$\mathcal{O}(\log n)$-platzbeschränkte TM bedeutet hier, dass das Arbeitsband beschränkt ist, nicht das Eingabe- und Ausgabeband.

Satz:

$$A \leq^{\log} B, B \leq^{\log} C \Rightarrow A \leq^{\log} C.$$

Definition: Sei $\mathcal{C}$ eine Komplexitätsklasse (z.B. NP, P, ...). B ist $\mathcal{C}$-*vollständig via logspace-Reduktion*, wenn
- $B \in \mathcal{C}$ und
- $A \leq^{\log} B$ für alle $A \in \mathcal{C}$.

Korrollar zum Beweis von obigem Satz: SAT ist NP-vollständig via logspace-Reduktion.

5.3 Varianten von SAT

Komplexität von Teilklassen van AL.

Zur Erinnerung: Jede Formel aussagenlogische Formel ψ ist äquivalent zu einer Formel ψ_D in DNF und ψ_K in KNF.

Es gilt:

$$\psi \equiv \psi_D := \bigvee_{\substack{\epsilon:\, V(\psi)\to\{0,1\} \\ \epsilon(\psi)=1}} \bigwedge_{x\in V(\psi)} x^\epsilon$$

und

$$\psi \equiv \psi_K := \bigwedge_{\substack{\epsilon:\, V(\psi)\to\{0,1\} \\ \epsilon(\psi)=0}} \bigvee_{x\in V(\psi)} \neg x^\epsilon,$$

wobei $x^\epsilon = \begin{cases} x & \epsilon(x) = 1 \\ \neg x & \epsilon(x) = 0 \end{cases}$.

Betrachte

$$\text{SAT-DNF} := \{\psi \in AL \mid \psi \text{ in DNF und } \psi \text{ erfüllbar}\},$$

$$\text{SAT-KNF} := \{\psi \in AL \mid \psi \text{ in KNF und } \psi \text{ erfüllbar}\}.$$

Satz: SAT-DNF ist in P, sogar in LOGSPACE.

Satz: SAT-KNF ist NP-vollständig.

Beweis: Übung.

Horn-Formeln

Eine *Horn-Formel* ist eine Formel der Form $\psi = \bigwedge_i \bigvee_j y_{i,j}$ in KNF, wobei jedes Konjunkt $\bigvee_j y_{i,j}$ höchstens *ein* positives Literal enthält.

$$\neg x_1 \vee \cdots \vee \neg x_n \vee x \equiv x_1 \wedge \cdots \wedge x_n \to x$$
$$\neg x_1 \vee \cdots \vee \neg x_n \equiv x_1 \wedge \cdots \wedge x_n \to 0$$

$$\text{HORN-SAT} := \{\psi \in AL \mid \psi \text{ erfüllbare Horn-Formel}\}.$$

Satz: HORN-SAT ist P-vollständig via logspace-Reduktion.

Beweis: Übung.

Definition: Eine Formel ist in r-KNF ($r \in \mathbb{N}$), wenn sie die Form

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} Y_{i,j}$$

besitzt, mit $m_i \leq r$ für alle i.
[Jede Formel hat höchstens r Literale.]

Satz: 3SAT ist NP-vollständig.

Beweis: Übung.

Gilt auch für r-SAT, wenn $r \geq 3$.

Komplexität von 2SAT?

Satz: 2SAT ist in P.

Man kann sogar zeigen, dass 2SAT in NLOGSPACE ist.

6 Probabilistische Turingmaschinen und Komplexiätsklassen

Für $m \in \mathbb{N}$ betrachte $\{0,1\}^m$ als Wahrscheinlichkeitsraum mit Gleichverteilung, d.h. für jedes $u \in \{0,1\}^m$ gilt

$$\Pr_{y \in \{0,1\}^m}[y = u] = \frac{1}{2^m}.$$

Definition: Eine *probabilistische Turingmaschine* (PTM) ist eine TM mit der Eingabe $(x, y) \in \Sigma^* \times \{0,1\}^*$, wobei $x \in \Sigma^*$ die eigentliche Eingabe ist und $y \in \{0,1\}^*$ ein Zufallswort zur Steuerung der Berechnung ist.

Definition: Eine PTM M ist $p(n)$-*zeitbeschränkt*, wenn M auf jede Eingabe (x, y) höchstens $p(|x|)$ viele Schritte ausführt.

Bei einer $p(n)$-zeitbeschränkten PTM M kann man ohne Einschränkung fordern, dass $|y| \leq p(|x|)$ ist, da M höchstens $p(|x|)$ Bits von y lesen kann.

Definition: Ist $p(n)$ ein Polynom und M eine $p(n)$-zeitbeschränkte PTM M, dann nennen wir M eine *probabilistische Polynomzeit-TM* (PPTM).

Sei M $p(n)$-zeitbeschränkt, $E(M) \subseteq \Sigma^* \times \{0,1\}^*$ die von M als Akzeptor über $\Sigma^* \times \{0,1\}^*$ akzeptierte Sprache. M ist also ein probabilistische Akzeptor über Σ^*. Sei $x \in \Sigma^*$, $|x| = n$. Dann ist die W'keit, dass M x akzeptiert wie folgt definiert:

$$\begin{aligned}\Pr[M \text{ akzeptiert } x] &:= \Pr_{y \in \{0,1\}^{p(n)}}[(x, y) \in E(M)] \\ &= \frac{\left|\{y \in \{0,1\}^{p(n)} \mid (x, y) \in E(M)\}\right|}{2^{p(n)}}.\end{aligned}$$

Lemma: Sei $A \subseteq \Sigma^*$. Dann gilt $A \in \mathsf{NP}$ genau dann, wenn es eine PPTM M gibt, mit

$$A = \{x \in \Sigma^* \mid \Pr_{(x,y) \in E(M)}[M \text{ akz. } x] > 0\}.$$

Beweis: Trivial, nach der Definition von NP. □

Wir hätten gerne eine Komplexitätsklasse, für die Raten von y "gute" Ergebnisse liefert, d.h. falls $x \in A$, dann sollte $\Pr[M \text{ akz. } x]$ klein sein. Wir

erwarten also eine geringe *Irrtumswarscheinlichkeit* (IW) (oder Fehler-). Es gibt zwei Arten von Fehlern:

a) *falschpositiv:* Es gilt $x \notin A$, aber $\Pr[M \text{ akz. } x] > 0$. $\Pr[M \text{ akz. } x]$ ist dann die IW für falschpositiv.

b) *falschnegativ:* Es gilt $x \in A$, aber $\Pr[M \text{ akz. } x] < 1$. $1 - \Pr[M \text{ akz. } x]$ ist dann die IW für falschnegativ.

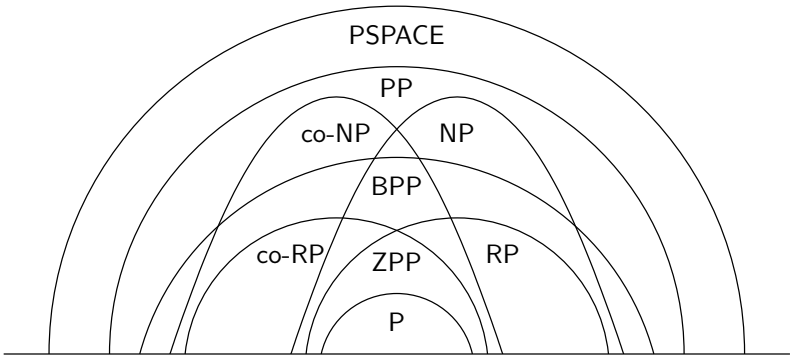
Definition: Wir definieren jetzt verschiedene Komplexitätsklassen:

- PP (probabilistic polynomial time): $A \in \mathsf{PP}$, wenn eine PPTM M existiert mit
$$A = \{x \mid \Pr[M \text{ akz. } x] > \frac{1}{2}\}.$$
- BPP (bounded error probabilistic polynomial time): $A \in \mathsf{BPP}$, wenn eine PPTM M existiert mit
$$x \in A \Longrightarrow \Pr[M \text{ akz. } x] \geq \frac{2}{3} \text{ und}$$
$$x \notin A \Longrightarrow \Pr[M \text{ akz. } x] \leq \frac{1}{3}.$$
- RP (random polynomial time): $A \in \mathsf{RP}$, wenn eine PPTM M existiert mit
$$x \in A \Longrightarrow \Pr[M \text{ akz. } x] \geq \frac{2}{3} \text{ und}$$
$$x \notin A \Longrightarrow \Pr[M \text{ akz. } x] = 0.$$
- $\mathsf{co\text{-}RP}$: $A \in \mathsf{co\text{-}RP} \Longleftrightarrow \overline{A} \in \mathsf{RP}$. D.h. es gibt eine PPTM M mit
$$x \in A \Longrightarrow \Pr[M \text{ akz. } x] = 1 \text{ und}$$
$$x \notin A \Longrightarrow \Pr[M \text{ akz. } x] \leq \frac{1}{3}.$$
- ZPP (zero error probabilistic polynomial time):
$$\mathsf{ZPP} := \mathsf{RP} \cap \mathsf{co\text{-}RP}.$$

Bemerkungen:

- PP: Die IW ist für beide Arten jeweils $\frac{1}{2}$.
- BPP: Die IW ist für beide Arten jeweils $\frac{1}{3}$.
- RP: Die IW ist für falschpositiv 0 und für falschnegativ $\frac{1}{3}$.
- co-RP: Die IW ist für falschpositiv $\frac{1}{3}$ und für falschnegativ 0.
- ZPP: Anschaulich kann man sich diese Klasse vorstellen als die Probleme, die in erwarteter polynomieller Zeit entschieden werden können. Man kann die Algorithmen für RP und co-RP gleichzeitig laufen lassen und man ist fertig, wenn beide das selbe Ergebniss liefern, da sich einer auf keinen Fall irrt. Die Laufzeit ist dann erwartet polynomiell (siehe dazu die Übung). Man nennt diese Algorithmen auch die *Las-Vegas-Algorithmen*.

Beziehungen zwischen den Komplexitätsklassen:



Satz: Es gilt NP $\subseteq$ PP $\subseteq$ PSPACE.

Beweis: Wir zeigen zunächst NP $\subseteq$ PP. Sei dazu $A \in$ NP. Nach obigem Lemma gibt es eine PPTM M mit $A = \{x \mid \Pr[M \text{ akz. } x] > 0\}$. Wir konstruieren jetzt eine neue PPTM M' wie folgt: M' akzeptiert $(x, y_0y_1y_2\dots)$ genau dann, wenn $y_0 = 1$ oder M akzeptiert $(x, y_1y_2\dots)$. Dann gilt

$$\Pr[M' \text{ akz. } x] = \frac{1}{2} + \Pr[M \text{ akz. } x] > \frac{1}{2}.$$

Also gilt $A \in$ PP.

Wir zeigen jetzt PP $\subseteq$ PSPACE. Dazu probieren wir alle möglichen y der Länge $p(n)$ aus und zählt, wie oft akzeptiert wird und wie oft verworfen. Wenn über die Hälfte akzeptiert wird, so akzeptiert man auch. Für den Zähler braucht man nur $p(n)$ viel Platz. □

Der Satz gilt auch für co-NP, d.h.:

Satz: Es gilt co-NP $\subseteq$ PP $\subseteq$ PSPACE.

Reduktion der Irrtumswahrscheinlichkeit von BPP-Algorithmen

Idee: Wir führen ein Majoritätsvotum nach k-maliger Wiederholung durch.

Sei M eine $p(n)$-zeitbeschränkte PTM mit IW $\leq \epsilon < \frac{1}{2}$. Wir definieren nun M^k: M^k akzeptiert $(x, y_1 y_2 \dots y_k)$, $y_i \in \{0,1\}^{p(n)}$ genau dann, wenn $|\{i \mid (x, y_i) \in E(M)\}| \geq \frac{k}{2}$, wobei $n = |x|$. M^k ist eine PPTM, falls $k = k(n)$ ein Polynom in n ist.

$X_1, \dots, X_k$ seinen $\{0,1\}$-wertige Zufallsvariablen. Sei $p = \Pr[X_i = 1]$ für alle i und ein p. Dann ist $\Pr[X_i = 0] = 1 - p$ und wir nennen die X_i Bernulli Zufallsvariablen. $X := \sum_{i=1}^{k} X_i$ ist eine $\mathbb{N}$-wertige Zufallsvariable mit Binomialverteilung. Es gilt $\mathbb{E}[X] = k \cdot p$ und $\mathrm{Var}[X] = k \cdot p(1-p)$.

Lemma (Chernoff): Für jedes d gilt

$$\Pr[X - \underbrace{p \cdot k}_{\mathbb{E}[X]} \geq d] \leq e^{-\frac{d^2}{4kp(1-p)}} \overset{p(1-p)\leq\frac{1}{4}}{\leq} e^{-\frac{d^2}{k}}$$

und analog

$$\Pr[pk - X \geq d] \overset{p(1-p)\leq\frac{1}{4}}{\leq} e^{-\frac{d^2}{k}}$$

Anwendung des Chernoff Lemmas:

$\bar{y} = y_1 \dots y_k$, $y_i \in \{0,1\}^{p(n)}$. $X_i(\bar{y}) = \begin{cases} 1 & (x, y_i) \in E(M) \\ 0 & \text{sonst} \end{cases}$. Sei A entscheidbar durch BPP-Algorithmus M mit IW $\leq \epsilon < \frac{1}{2}$. Sei $x \notin A$ und $p := \Pr[X_i = 1] \leq \epsilon$ und $X := \sum_{i=1}^{k} X_i$. Dann gilt

$$\Pr[M^k \text{ akz. } x] = \Pr[X \geq \frac{k}{2}].$$

Mit Chernoff folgt

$$\Pr[X \geq \frac{k}{2}] = \Pr[X - pk \geq (\frac{1}{2} - p)k] \leq e^{-(\frac{1}{2}-p)^2 k} \leq 2^{-\Omega(k)}.$$

Sei $q(n)$ ein beliebiges Polynom. Für $k \geq c \cdot q(n)$ (c geeignete Konstante) erhalten wir eine falsch-positive IW $\leq 2^{-q(n)}$.

Analog für falsch-negativ. Sei $x \in A$, dann gilt

$$\Pr[M \text{ akz. } x] = \Pr[X_i = 1] =: p \geq 1 - \epsilon.$$

Also

$$\Pr[M^k \text{ akz. } x \text{ nicht}] = \Pr[X \leq \frac{k}{2}] \leq \Pr[pk - X \geq (p - \frac{1}{2})k \leq e^{-(p-\frac{1}{2})^2 k} = 2^{-\Omega(k)}.$$

Dies zeigt:

Satz: Zu jedem $A \in \mathsf{BPP}$ und jedem Polynom $q(n)$ gibt es eine PPTM M, welche A mit IW $\leq 2^{-q(n)}$ akzeptiert, d.h. $x \in A \Rightarrow \Pr[M \text{ akz. } x] \geq 1 - 2^{-q(n)}$ und $x \notin A \Rightarrow \Pr[M \text{ akz. } x] \leq 2^{-q(n)}$ und

Analog für RP (sogar ohne Chernoff).

Folgerung: Sei M ein BPP-Algorithmus für A mit IW $\leq 2^{-q(n)}$. Für alle x gilt also

$$\Pr[M \text{ akz. } x] = \frac{\left|\{y \in \{0,1\}^{p(n)} \mid (x,y) \in E(M) \Leftrightarrow x \in A\}\right|}{2^{p(n)}} \geq 1 - 2^{-q(n)}.$$

Dann gilt, dass es für jedes feste hinreichend große n Zufallswerte $y \in \{0,1\}^{p(n)}$ gibt, die *für alle* x der Länge n das korrekte Ergebnis liefern.

Beweis:

$$\begin{aligned}&\left|\{y \in \{0,1\}^{p(n)} \mid y \text{ liefert falsches Ergebnis für min. ein } x \in \Sigma^n\}\right| \\ &\leq |\Sigma^n| \cdot 2^{-q(n)} \cdot 2^{p(n)}\end{aligned}$$

Wähle nun $q(n)$ so, dass $|\Sigma^n| \cdot 2^{-q(n)} \to 0$, $n \to \infty$. Dann gibt es ein n_0, so dass für alle $n \geq n_0$ gilt $|\Sigma^n|\, 2^{-q(n)} < 1$. Insbesondere gilt für alle $n \geq n_0$

$$\left|\{y \in \{0,1\}^{p(n)} \mid y \text{ liefert falsches Ergebnis für min. ein } x \in \Sigma^n\}\right| < 2^{p(n)}.$$

Es folgt, dass es für jedes große n ($\geq n_0$) ein $y(n) \in \{0,1\}^{p(n)}$ gibt, welches richtige Entscheidungen liefert *für alle* $x \in \Sigma^n$. Für $n \leq n_0$ Sonderbehandlung. D.h. es gibt eine Funktion $f\colon \mathbb{N} \to \{0,1\}^*$ mit

- f ist polynomiell beschränkt, d.h. $|f(n)| = p(n)$ für ein Polynom p und

- $A = \{x \mid (x, f(|x|)) \in E(M)\}$ (die Entscheidung, ob $(x, f(|x|)) \in E(M)$ is in polynomieller Zeit möglich.

Aber f ist im Allgemeinen nicht polynomiell berechenbar.

Definition: $A \subseteq \Sigma^*$ ist *nicht-uniform polynomiell entscheidbar* ($A \in \mathsf{P}/poly$), wenn eine Funktion $f\colon \mathbb{N} \to \{0,1\}^*$ und eine Menge $B \in \mathsf{P}$ existiert mit

- $|f(n)| \leq p(n)$, p Polynom und
- $A = \{x \in \Sigma^* \mid (x, f(|x|)) \in B\}$.

Beachte, dass f nicht berechenbar sein muss.

Aus obigem Satz folgt, dass

$$\mathsf{BPP} \subseteq \mathsf{P}/poly.$$

Beispiel: Ordne jeder Inputlänge einen Schaltkreis polynomieller Länge zu, welcher $A \cap \Sigma^n$ entscheidet.