Bemerkung 4.6)

Die erste Inklusion von Fakt 4.6 besagt, dass

$$DTIME(S(n)) \subseteq SPACE(S(n)).$$

Es ist sogar ein besseres Ergebnis bekannt (hier ohne Beweis), das besagt.

$$DTIME(S(n)) \subseteq SPACE\left(\frac{S(n)}{\log(S(n))}\right)$$

(für hinreichend "gutartige" Funktionen S);
siehe Hopcroft, Paul, Valiant (1975).

Definition 4.7 (Platzklassen)

$$PSPACE := \bigcup_{c>0} SPACE(n^c) \qquad NPSPACE := \bigcup_{c>0} NSPACE(n^c)$$

$$L := SPACE(\log n) \qquad NL := NSPACE(\log n)$$

Die Klassen L und NL werden manchmal auch LOGSPACE und NLOGSPACE genannt.

$$POLYLOGSPACE := \bigcup_{c>0} SPACE((\log n)^c)$$

Beachte: Aus Satz 4.6 folgt, dass $NL \subseteq P$ und $NPSPACE \subseteq EXP$

Beispiele 4.8

(a) SAT $\in$ PSPACE:

Sei φ die eingegebene aussagenlogische Formel, sei k die Anzahl der aussagenlogischen Variablen, die in φ vorkommen

Eine polynomiell Platzbeschränkte TM, die entscheidet, ob φ erfüllbar ist, kann wie folgt vorgehen:

```
i := 0
while i < 2^k :
    schreibe die Binärrepräsentation von i der Länge k auf
    ein Arbeitsband und fasse diese als eine
    Belegung der k Variablen von φ auf
    teste, ob φ von dieser Belegung erfüllt wird
    Falls ja: STOPP mit Ausgabe 'ja'
    sonst: weiter mit i := i+1
```

Da SAT NP-vollständig ist und da PSPACE abgeschlossen ist unter Polynomialzeit-Reduktionen, erhält man so auch, dass **NP $\subseteq$ PSPACE**

(Details: Übung).

(b) Die beiden folgenden Sprachen gehören zur Komplexitätsklasse L:

$$EVEN := \{ x \in \{0,1\}^* : \text{die Anzahl der 1en in } x \text{ ist gerade} \}$$

$$MULT := \{ \langle a, b, c \rangle : a, b, c \in \mathbb{N},\ a \cdot b = c \}$$

Details: Übung!

(c) Die Sprache

$$PATH := \{ \langle G, s, t \rangle : G \text{ ist ein gerichteter Graph, in dem es einen Weg von Knoten } s \text{ zu Knoten } t \text{ gibt} \}$$

gehört zur Komplexitätsklasse NL

Beweis: Sei $G = (V, E)$ mit $n := |V|$.

Lem: • Falls es in G einen Weg von s nach t gibt, dann gibt es auch einen Weg von s nach t der Länge $\leq n$

• Jeder Knoten von G kann durch einen Bitstring der Länge $\log n$ repräsentiert werden

Eine $O(\log n)$-platzbeschränkte NDTM kann wie folgt ermitteln, ob es in G einen Weg von s nach t gibt.

```
x := s
i := 0
while i ≤ n
    rate einen beliebigen Knoten y
    teste, ob es in G eine Kante von x zu y gibt
    Falls nein: STOPP mit Ausgabe "nein"
    Sonst: Falls y=t, so STOPP mit Ausgabe "ja"
           Sonst: x := y, i := i+1
STOPP mit Ausgabe "nein"
```

Wir werden später sehen, dass PATH vollständig für die Klasse NL ist und daher vermutlich nicht in L liegt.
Ein relativ neues, überraschendes Resultat besagt, dass die Einschränkung von PATH auf ungerichtete Graphen tatsächlich in der Klasse L liegt (Omer Reingold, 2005)

Lineare Kompression und Platzhierarchiesatz

Ähnlich wie für zeitbeschränkte Berechnungen erhalten wir:

Satz 4.9 (Lineare Kompression)
Sei $S: \mathbb{N} \to \mathbb{N}$ und sei $\varepsilon > 0$.
Sei M eine $S(n)$-platzbeschränkte TM, die eine Sprache $L \subseteq \{0,1\}^*$ entscheidet.
Dann gibt es auch eine $\varepsilon \cdot S(n)$-platzbeschränkte 2-Band TM, die L entscheidet.

Beweis: Übung!

Theorem 4.10 (Platzhierarchiesatz von Stearns, Hartmanis, Lewis, 1965)
Seien $s, S: \mathbb{N} \to \mathbb{N}$ zwei platzkonstruierbare Funktionen mit $s(n) \geq \log n$ und $s(n) = o(S(n))$. Dann gilt:

$$\mathrm{SPACE}(s(n)) \subsetneq \mathrm{SPACE}(S(n))$$

Beweis: Analog zum Beweis des Zeithierarchiesatzes.
Details: Übung!
□

Folgerung 4.11

F.a. $c \geq 1$ gilt:

- $\mathrm{SPACE}\left((\log n)^c\right) \subsetneq \mathrm{SPACE}\left((\log n)^{c+1}\right)$
- $\mathrm{SPACE}\left(n^c\right) \subsetneq \mathrm{SPACE}\left(n^{c+1}\right)$

Insbes: $L \subsetneq \mathrm{POLYLOGSPACE} \subsetneq \mathrm{PSPACE}$

Der Satz von Savitch

Ein (vielleicht überraschendes) Ergebnis von Savitch (1970) besagt, dass nichtdeterministische Berechnungen platzeffizient durch deterministische Berechnungen simuliert werden können:

Theorem 4.12 (Satz von Savitch, 1970)

Für jedes platzkonstruierbare $S: \mathbb{N} \to \mathbb{N}$ mit $S(n) \geq \log n$ gilt:

$$\mathrm{NSPACE}(S(n)) \subseteq \mathrm{SPACE}\left(S(n)^2\right)$$

Folgerung 4.13

(a) $\mathrm{NPSPACE} = \mathrm{PSPACE}$
(man vergleiche dazu die offene Frage, ob $NP = P$!)

(b) $\mathrm{SPACE}(\log n) \overset{\text{Def}}{=} L \subseteq NL \subseteq \mathrm{SPACE}\left((\log n)^2\right)$

Beweis von Theorem 4.12:

Sei $L \in \mathrm{NSPACE}(S(n))$, und sei M eine $c \cdot S(n)$-platzbeschränkte TM, die L entscheidet (für eine

geeignete Konstante c).
Dann hat für jede Eingabe $x \in \{0,1\}^*$ der Konfigurationsgraph $G_{M,x}$ höchstens $N := 2^{d \cdot S(n)}$ Knoten, für $n := |x|$ und eine geeignete Konstante d — und jeder Knoten von $G_{M,x}$ wird durch einen Bitstring der Länge $d \cdot S(n)$ repräsentiert. Außerdem gilt:

$x \in L \iff$ in $G_{M,x}$ gibt es einen Weg von Knoten C_{start} zu Knoten C_{accept} der Länge $\leq N$.

Betrachte folgende rekursive Prozedur

REACH?(u,v,i),

die entscheidet, ob es in $G_{M,x}$ einen Weg der Länge $\leq 2^i$ von Knoten u zu Knoten v gibt:

REACH? (u,v,i)

Falls $i=0$:
  Falls $u=v$ oder es in $G_{M,x}$ eine Kante von u nach v gibt, so
    STOPP mit Antwort "ja"
  Sonst:
    STOPP mit Antwort "nein"

Falls $i>0$:
  Für jeden Knoten w von $G_{M,x}$ tue folgendes:
    Falls REACH? $(u,w,i-1)$ = "ja" und REACH? $(w,v,i-1)$ = "ja", so
      STOPP mit Antwort "ja"
  STOPP mit Antwort "nein".

Per Induktion nach i erhält man, dass REACH? (u,v,i) genau dann die Antwort "ja" liefert, wenn es in $G_{M,x}$ einen Weg der Länge $\leq 2^i$ von u nach v gibt.

Der Platzbedarf $f(i)$ von REACH bei Eingabe u, v, i ist:

- $4 \cdot d \cdot S(n) + 2$ Bit zum Speichern von u, v, i, w sowie zum Speichern der Antwort "ja"/"nein" von REACH?$(u, w, i-1)$ und REACH?$(w, v, i-1)$,

plus

- $f(i-1)$ Platz, der zum Berechnen von REACH?$(u', v', i-1)$ (zuerst für $(u', v') = (u, w)$, dann für $(u', v') = (w, v)$) nötig ist.

Wir erhalten die Rekursionsgleichung

$$f(i) = 4 \cdot d \cdot S(n) + 2 + f(i-1)$$

Für $i_{max} := \log N = d \cdot S(n)$ erhalten wir

$$f(i_{max}) = d \cdot S(n) \cdot (4 d S(n) + 2) = O(S(n)^2)$$

Insgesamt erhalten wir:

REACH?$(C_{start}, C_{accept}, i_{max})$

entscheidet auf Platz $O(S(n)^2)$, ob $x \in L$ liegt.

Somit ist $L \in SPACE(S(n)^2)$. □

Bemerkung: Eine Analyse des Zeitverbrauchs zeigt, dass REACH?$(C_{start}, C_{accept}, i_{max})$ bis zu $\geq 2^{(S(n)^2)}$ viele Schritte verbrauchen kann (Details: Übung!). Daher liefert uns dieses Algorithmus keinen neuen Beweis für die in Satz 4.6 gezeigte Aussage $NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$.

4.2 PSPACE-Vollständigkeit

Wir wissen bereits, dass $P \subseteq NP \subseteq PSPACE = NPSPACE$ gilt.

Da vermutet wird, dass $P \neq NP$ ist, ist vermutlich auch $P \neq PSPACE$ — bewiesen kann das bisher aber niemand.

Die "schwierigsten" Probleme in PSPACE sind die PSPACE-vollständigen Probleme:

Definition 4.14: Sei $L' \subseteq \{0,1\}^*$

(a) L' ist PSPACE-hart, falls für jedes $L \in PSPACE$ gilt: $L \leq_p L'$.

(b) L' ist PSPACE-vollständig, falls $L' \in PSPACE$ und L' PSPACE-hart ist.

Bemerkung 4.15

Analog zum Begriff der NP-Vollständigkeit gilt:

(a) Falls es eine PSPACE-vollständige Sprache L' mit $L' \in P$ gibt, so ist $P = PSPACE$.

(b) Die folgende Sprache ist PSPACE-vollständig:

$$SPACETMSAT := \{ \langle \alpha, x, 1^s \rangle : \alpha, x \in \{0,1\}^*, s \in \mathbb{N},$$
DTM M_α akzeptiert Eingabe x und nutzt dabei auf jedem ihrer Arbeitsbänder $\leq s$ Zellen $\}$.

Beweis: Übung!

Wir werden nun zeigen, dass die folgende Verallgemeinerung des SAT-Problems PSPACE-vollständig ist

Definition 4.16 (Quantifizierte Boolesche Formeln; TQBF)

(a) Eine quantifizierte Boolesche Formel (kurz: QBF) ist eine Formel der Form

$$Q_1 x_1 \; Q_2 x_2 \cdots Q_n x_n \; \varphi(x_1, \ldots, x_n)$$

wobei
- $n \in \mathbb{N}$,
- $\varphi(x_1, \ldots, x_n)$ ist eine aussagenlogische Formel über den Variablen $x_1, \ldots, x_n$
- $Q_i \in \{\exists, \forall\}$ (f.a. $i \in \{1, \ldots, n\}$).

(b) Eine QBF Φ der Form $Q_1 x_1 \, Q_2 x_2 \cdots Q_n x_n \; \varphi(x_1, \ldots, x_n)$ ist <u>wahr</u>, falls die Aussage

"$Q_1 z_1 \in \{0,1\} \; Q_2 z_2 \in \{0,1\} \cdots Q_n z_n \in \{0,1\}$:
φ wird von der Belegung, die die Variablen $x_1, \ldots, x_n$ mit den Worten $z_1, \ldots, z_n$ belegt, erfüllt"

wahr ist;
ansonsten ist Φ <u>falsch</u>

<u>Beispiele:</u>
- $\forall x_1 \exists x_2 \big((x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)\big)$ ist wahr
- $\exists x_1 \forall x_2 \big((x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)\big)$ ist falsch
- $\varphi(x_1, \ldots, x_n)$ erfüllbar $\Leftrightarrow$ $\exists x_1 \exists x_2 \cdots \exists x_n \, \varphi(x_1, \ldots, x_n)$ ist wahr
- $\varphi(x_1, \ldots, x_n)$ allgemeingültig $\Leftrightarrow$ $\forall x_1 \forall x_2 \cdots \forall x_n \, \varphi(x_1, \ldots, x_n)$ ist wahr

(c) Das Entscheidungsproblem TQBF ist wie folgt definiert:

$$TQBF := \{ \Phi : \Phi \text{ ist eine } \underline{\text{wahre}} \text{ QBF} \}.$$

(T steht für "True")

Theorem 4.17 (Satz von Stockmeyer und Meyer, 1973)

TQBF ist PSPACE-vollständig.

Beweis:

TQBF $\in$ PSPACE: Übung!

PSPACE-Härte von TQBF:

Sei $L \in$ PSPACE, und sei M eine (det.) TM, die L mit Platz $S(n)$ entscheidet, für ein Polynom $S(n)$.

Zu zeigen: $L \leq_p$ TQBF.

Dazu konstruieren wir für jede Eingabe $x \in \{0,1\}^*$ und $n := |x|$ eine QBF $\Phi_{M,x}$ der Größe $O(S(n)^2)$, so dass gilt:

$\Phi_{M,x}$ ist wahr $\Leftrightarrow$ in $G_{M,x}$ gibt es einen Weg von C_{start} zu C_{accept}

Gemäß Fakt 4.5(b) gibt es eine CNF-Formel $\varphi_{M,x}(\bar{y}, \bar{y}')$, die für Belegungen von $\bar{y}, \bar{y}'$, die zwei Knoten C, C' von $G_{M,x}$ repräsentieren, genau dann

erfüllt ist, wenn es in $G_{n,t}$ eine Kante von C zu C' gibt. Die Länge von $\varphi_{n,t}$ ist $\sigma(S(n))$

Ziel: Konstruiere, für jedes $i \geq 0$, eine Formel $\psi_i(\bar{y}, \bar{y}')$, die besagt:
es gibt in $G_{n,t}$ einen Weg der Länge $\leq 2^i$ von C nach C'.

Idee:

- $\psi_0(\bar{y}, \bar{y}') := \left(\bar{y} = \bar{y}' \vee \varphi_{n,t}(\bar{y}, \bar{y}') \right)$, wobei
$\bar{y} = \bar{y}'$ eine Abkürzung ist für $\bigwedge_{i=1}^{d \cdot S(n)} \left((y_i \wedge y_i') \vee (\neg y_i \wedge \neg y_i') \right)$

- $\psi_{i+1}(\bar{y}, \bar{y}') := \exists \bar{y}'' \left(\psi_i(\bar{y}, \bar{y}'') \wedge \psi_i(\bar{y}'', \bar{y}') \right)$

Problem: Beim Übergang von ψ_i zu ψ_{i+1} verdoppelt sich die Größe der Formel. Die Länge von ψ_i ist daher $> 2^i$
Da $G_{n,t}$ aus $2^{d \cdot S(n)}$ Knoten besteht, und wir Wege bis zu dieser Länge berücksichtigen müssen, hat die Formel $\psi_{d \cdot S(n)}$ die Länge $> 2^{d \cdot S(n)}$ und kann nicht von einer Polynomialzeit-Reduktion erzeugt werden.

Lösung: Wähle eine kompaktere Formel ψ_{i+1}, die nur 1x (statt 2x) auf ψ_i zurückgreift.
Die Länge von ψ_i ist dann polynomiell in i und damit hat

$\psi_{d \cdot S(n)}$ die Länge $\text{poly}(S(n))$ und kann von einer Polynomialzeit-Reduktion konstruiert werden.

Konkret wählen wir:

$$\psi_{i+1}(\bar{y}, \bar{y}') := \exists \bar{y}'' \; \forall \bar{u} \; \forall \bar{u}' \; \Big(\big((\bar{u} = \bar{y} \wedge \bar{u}' = \bar{y}'') \vee (\bar{u} = \bar{y}'' \wedge \bar{u}' = \bar{y}') \big) \rightarrow \psi_i(\bar{u}, \bar{u}') \Big)$$

Dann gilt:

- Länge von ψ_0: $O(S(n))$
- Länge von ψ_{i+1}: $O(S(n))$ + Länge von ψ_i

Insgesamt ergibt sich dann:

- Länge von ψ_i: $O(i \cdot S(n))$.

Wir wählen $\Phi_{n,x} := \psi_{d \cdot S(n)}(C_{\text{start}}, C_{\text{accept}})$

und erhalten damit eine QBF-Formel, die genau dann wahr ist, wenn es in $G_{n,x}$ einen Weg von C_{start} zu C_{accept} gibt (d.h. wenn x von M akzeptiert wird).

Die Formel $\Phi_{n,x}$ hat die Länge $O(S(n)^2)$ und kann in Zeit $\text{poly}(S(n)) = \text{poly}(n)$ berechnet werden.

Somit ist $L \leq_p \text{TQBF}$. □

Bemerkung 4.18:

Der obige Beweis funktioniert genauso, wenn M eine NDTM (statt einer DTM) ist und zeigt somit, dass TQBF sogar NPSPACE-hart ist.
Aus TQBF $\in$ PSPACE folgt dann PSPACE = NPSPACE, ohne dass wir den Satz von Savitch anwenden müssen (vgl. Theorem 4.12 und Folgerung 4.13).

Bemerkung 4.19:

Die Frage, ob eine QBF Φ der Form

$$\exists x_1 \, \forall x_2 \, \exists x_3 \, \dots \, \forall x_{2n} \, \varphi(x_1, \dots, x_{2n})$$

zu TQBF gehört, können wir als ein 2-Personen-Spiel auffassen:

Spieler 1 wählt eine Belegung der Variablen x_1, danach antwortet Spieler 2 mit einer Belegung der Variablen x_2, dann wählt wieder Spieler 1 eine Belegung der Variablen x_3 usw., und am Ende hat Spieler 1 genau dann gewonnen, wenn die während des Spiels erzeugte Belegung die Formel $\varphi(x_1, \dots, x_{2n})$ erfüllt.

Man sieht leicht, dass Spieler 1 genau dann eine Gewinnstrategie für dieses Spiel hat, wenn die QBF-Formel Φ wahr ist, d.h. zu TQBF gehört.

Die PSPACE-Vollständigkeit von TQBF zeigt also, dass das Finden von Gewinnstrategien für 2-Personen-Spiele gewissermaßen die "Essenz der Klasse PSPACE" beschreibt.

Man kann beispielsweise auch zeigen, dass verallgemeinerte Versionen von Schach oder Go (auf Spielbrettern der Größe $n \times n$) PSPACE-vollständig sind.

4.3 Die Klasse NL

Zur Erinnerung:

- $L \overset{\text{Def}}{=} \text{SPACE}(\log n)$ "Logspace"
- $NL \overset{\text{Def}}{=} \text{NSPACE}(\log n)$ "ndet. Logspace"
- $L \subseteq NL \underset{\text{Satz 4}}{\subseteq} \text{SPACE}((\log n)^2)$ und $NL \subseteq P$

NL-Vollständigkeit

Wir wollen den Begriff der NL-Vollständigkeit so definieren, dass gilt:

Liegt ein NL-vollständiges Problem L' in der Klasse L, so ist $L = NL$.

Dazu benötigen wir einen Reduktionsbegriff $\leq_\ell$, unter dem die Klasse L abgeschlossen ist, d.h. für den gilt

Ist $L_1 \leq_\ell L_2$ und $L_2 \in L$, so auch $L_1 \in L$.

Für Polynomialzeit-Reduktionen ist das anscheinend nicht der Fall, d.h. wir können nicht beweisen, dass gilt: Ist $L_1 \leq_p L_2$ und $L_2 \in L$, so auch $L_1 \in L$

Definition 4.20 (Logspace-Reduktionen)

Sei $f: \{0,1\}^* \to \{0,1\}^*$

(a) f heißt write-once logspace-berechenbar, falls es eine TM gibt, die f berechnet und dabei

- ein "write-once" Ausgabeband benützt, auf das sie in jedem Schritt entweder ein Symbol schreibt und dann nach rechts geht oder kein Symbol schreibt und den Kopf an der aktuellen Position lässt, und weiter
- Arbeitsbänder, auf denen sie nur $O(\log n)$ Zellen benützt

(b) f heißt implizit logspace-berechenbar, falls gilt.

- f ist polynomiell beschränkt, d.h. $|f(x)| \leq \text{poly}(|x|)$, f.a. $x \in \{0,1\}^*$, und
- die beiden Sprachen
  $L_f := \{ \langle x, i \rangle : x \in \{0,1\}^*, i \in \mathbb{N}, f(x)_i = 1 \}$ und
  $L'_f := \{ \langle x, i \rangle : x \in \{0,1\}^*, i \in \mathbb{N}, i \leq |f(x)| \}$

  liegen in der Klasse L.

Fakt 4.21

$f: \{0,1\}^* \to \{0,1\}^*$ ist genau dann implizit logspace-berechenbar, wenn f write-once logspace-berechenbar ist.

Beweis: Übung!

Definition 4.22 (logspace-Reduktionsbarkeit)

Seien $B, C \subseteq \{0,1\}^*$.
B ist logspace-reduzierbar auf C (kurz: $B \leq_e C$), falls es eine implizit logspace-berechenbare Funktion $f: \{0,1\}^* \to \{0,1\}^*$ gibt, so dass f.a. $x \in \{0,1\}^*$ gilt:

$$x \in B \iff f(x) \in C.$$

Lemma 4.23

(a) $\leq_e$ ist transitiv, d.h. f.a. $B, C, D \subseteq \{0,1\}^*$ gilt:
Falls $B \leq_e C$ und $C \leq_e D$, so $B \leq_e D$.

(b) L ist abgeschlossen unter logspace-Reduktionen, d.h.
f.a. $B, C \subseteq \{0,1\}^*$ gilt:
Falls $B \leq_e C$ und $C \in L$, so auch $B \in L$.

Beweis:
Seien $f, g: \{0,1\}^* \to \{0,1\}^*$ implizit logspace-berechenbar durch TMen M_f, M'_f, M_g, M'_g (für L_f, L'_f, L_g, L'_g).