Eine Sprache $U \subseteq \{0,1\}^*$ heißt unär, falls $U \subseteq \{1\}^*$ ist.

Theorem 2.22 (Der Satz von Berman, 1978)

Falls es eine unäre NP-vollständige Sprache gibt, so ist $P = NP$.

Beweis: siehe Übungsaufgaben.

3.2 Der Satz von Ladner

Theorem 3.6 (Der Satz von Ladner, 1975)

Falls $P \neq NP$, so gibt es eine Sprache L mit:

$L \in NP$, $L \notin P$ und L ist nicht NP-vollständig.

Beweis: Diagonalisierung!

Für jedes $H: \mathbb{N} \to \mathbb{N}$ sei SAT_H die folgendermaßen "aufgefüllte" ("gepaddete") Version von SAT:

$$SAT_H := \{ \psi 0 1^{(m^{H(m)})} : \psi \in SAT, m := |\psi| \}$$

Sei nun $H: \mathbb{N} \to \mathbb{N}$ wie folgt rekursiv definiert:

$H(0) := H(1) := 1$,

$H(n) :=$ die kleinste Zahl $i < \log\log n$, so dass f.a. $x \in \{0,1\}^*$ mit $|x| < \log n$ gilt:

M_i gibt bei Eingabe x nach $\leq i \cdot |x|^i$ Schritten den Wert $SAT_H(x)$ aus (d.h. 1 falls $x \in SAT_H$; 0 sonst)

— bzw. den Wert $\log\log n$, falls kein solches $i < \log\log n$ existiert.

Beachte: H ist wohldefiniert, da bei der Wahl von $H(n)$ nur auf SAT_H-Instanzen der Länge $< n$ betrachtet werden.

Behauptung 0:
Es gibt einen det. Polynomialzeit-Algorithmus, der bei Eingabe eines Worts der Länge n in Zeit $poly(n)$ die Zahl $H(n)$ berechnet.

Beweis: Übung!

Unter Verwendung von Beh 0 sieht man leicht, dass $SAT_H \in NP$ ist.

Behauptung 1:

(a) $SAT_H \in P \Rightarrow$ es gibt ein $C \in \mathbb{N}$ s.d. $H(n) \leq C$ f.a. $n \in \mathbb{N}$.

(b) $SAT_H \notin P \Rightarrow H(n) \xrightarrow{n \to \infty} \infty$.

Beweis:

(a): Sei $SAT_H \in P$ und sei M eine (det.) TM, die SAT_H in $\leq d \cdot n^d$ Schritten löst, für eine Konstante d.

Sei $i > d$ s.d. $M = M_i$.

Gemäß der Definition von H gilt f.a. $n > 2^{2^i}$, dass $H(n) < i$.

Daher fertig mit $C := \max\{i, H(0), H(1), \dots, H(2^{2^i})\}$

(b): Angenommen, $H(n) \not\xrightarrow{n \to \infty} \infty$.

Dann gibt es ein $C \in \mathbb{N}$ und eine unendliche Folge von Zahlen $0 \leq n_0 < n_1 < n_2 < \cdots$ s.d. f.a. $j \in \mathbb{N}$ gilt: $H(n_j) \leq C$.

Dann gibt es auch ein $i \leq C$ s.d. $H(n_j) = i$ für unendlich viele $j \in \mathbb{N}$ gilt.

Gemäß unserer Wahl von H gilt dann für jedes solche j:

F.a. $x \in \{0,1\}^*$ mit $|x| < \log n_j$ gibt M_i bei Eingabe x nach $\leq i \cdot |x|^i$ Schritten den Wert $SAT_H(x)$ aus.

Daher entscheidet M_i die Sprache SAT_H in $i \cdot n^i$ Schritten — d.h.: $SAT_H \in P$

□ Beh 1.

Behauptung 2:
Falls $P \neq NP$, so ist $SAT_H \notin P$

Beweis:
Angenommen, $SAT_H \in P$. Gemäß Beh 1(a) gibt es dann ein $C \in \mathbb{N}$ s.d. $H(n) \leq C$ f.a. $n \in \mathbb{N}$

Somit ist

$$SAT_H = \{ \varphi 0 1^{(n^{H(n)})} : \varphi \in SAT,\ n = |\varphi|,\ H(n) \leq C \}$$

Dann gibt es eine Polynomialzeit-Reduktion von SAT auf SAT_H, die bei Eingabe φ einfach $n = |\varphi|$ ermittelt, $H(n)$ berechnet (vgl. Beh. 0), $1^{(n^{H(n)})}$ berechnet (das geht in Zeit poly(n), da $H(n) \leq C$) und dann $\varphi 0 1^{(n^{H(n)})}$ ausgibt.
Aus $SAT_H \in P$ folgt dann: $SAT \in P$.
Da SAT NP-vollständig ist, gilt dann $P = NP$ □ Beh 2

Behauptung 3
Falls $P \neq NP$, so ist SAT_H nicht NP-vollständig.

Beweis: Es sei $P \neq NP$.
- Angenommen, SAT_H ist NP-vollständig. Dann gibt es eine Polynomialzeit-Reduktion f von SAT auf SAT_H. Sei $i \in \mathbb{N}$ s.d. f in $\leq n^i$ Schritten berechnet wesden kann.

- Außerdem wissen wir aus Beh 2 bereits, dass $SAT_H \notin P$ ist — und mit Beh 1(b) folgt daher, dass $H(n) \xrightarrow{n\to\infty} \infty$. Insbes. ist $H(n) > i$ für alle hinreichend großen n.

- Die Polyn.zeit-Reduktion f bildet eine SAT-Instanz φ der Länge n auf eine SAT_H-Instanz $\psi 01^{(m^{H(m)})}$ ab, wobei $m := |\psi|$ ist.

  Insbes. gilt:

  $m^{H(m)} < n^i$, da $f(\varphi)$ in n^i Schritten berechnet werden kann.

  Für alle hinreichend großen n muss dann

  $m < n^{1/2}$ sein

  (denn ansonsten würde für hinreichend große n mit $\frac{1}{2}H(m) > i$ gelten: $m^{H(m)} \geq \left(n^{1/2}\right)^{H(n^{1/2})} = n^{\frac{1}{2}\cdot H(m)} > n^i$ ↯).

  Somit gilt f.a. hinreichend großen n:
  f bildet SAT-Instanzen φ der Länge n ab auf SAT_H-Instanzen $\psi 01^{(m^{H(m)})}$, wobei $|\psi| = m < n^{1/2}$, s.d. gilt:

  $\varphi \in SAT \iff \psi \in SAT$.

- Durch wiederholtes Anwenden von f erhält man eine Folge $\psi_1, \psi_2, \psi_3, \ldots$ von SAT-Instanzen,

für die gilt:

(1) $\varphi \in SAT \iff \psi_1 \in SAT \iff \psi_2 \in SAT \iff \psi_3 \in SAT \iff \dots$

und

(2) $|\psi_1| < n^{\frac{1}{2}}$, $|\psi_2| < (n^{\frac{1}{2}})^{\frac{1}{2}}$, $|\psi_3| < n^{\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}}$, und allgemein f.a. j: $|\psi_j| < n^{(1/2^j)}$

Frage: Für welches j gilt: $n^{(1/2^j)} \leq \log n$?

Antwort:

$$n^{1/2^j} \leq \log n \iff 2^{(\log n)\cdot\frac{1}{2^j}} \leq 2^{\log\log n}$$

$$\iff (\log n)\cdot\frac{1}{2^j} \leq \log\log n$$

$$\iff \frac{\log n}{\log\log n} \leq 2^j \impliedby \log\log n \leq j$$

Somit gilt: $\log\log n$-faches iteriertes Anwenden der Polynomialzeit-Reduktion f liefert eine SAT-Instanz $\psi' := \psi_{\log\log n}$ der Länge $\leq \log n$,

s.d. $\varphi \in SAT \iff \psi' \in SAT$.

In ψ' können höchstens $|\psi'| \leq \log n$ verschiedene Variablen vorkommen. Ein det. Algorithmus, der nacheinander alle $2^{\log n} = n$ Variablenbelegungen durchprobiert, kann in poly(n) Schritten ermitteln ob $\psi' \in SAT$ ist. Zum Erzeugen von ψ' werden auch nur poly(n) Schritte benötigt, da

die in Zeit n^i berechenbare Reduktion insgesamt nur $\log\log n$ mal aufgerufen wird, um $\psi^i = \psi_{\log\log n}$ zu erzeugen.

Damit erhalten wir einen det. Polynomialzeit-Algorithmus, der SAT löst. Also $SAT \in P$ und daher $P = NP$.

↳ Widerspruch zur Annahme, dass $P \neq NP$ ist

□ Beh 3

Insgesamt ist für $L := SAT_{\#}$ damit Theorem 3.6. bewiesen

□

3.3 Orakel-TMen und die Grenzen der Diagonalisierung

Definition 3.7

a) Eine Orakel-TM ist eine (det.) TM, die ein spezielles sog. Orakel-Band besitzt sowie drei spezielle Zustände q_{query}, q_{yes}, q_{no}.
Bei Eingabe $x \in \{0,1\}^*$ mit Orakel $O \subseteq \{0,1\}^*$ geht M wie eine herkömmliche TM vor. Aber jedesmal, wenn M im Zustand q_{query} ist, ist sie im nächsten Schritt im Zustand q_{yes} bzw q_{no}, je nach dem ob das Wort, das gerade auf dem Orakel-Band

steht, ihr Sprache O gehört oder nicht.

(Beachte: Unabhängig davon, wie O aussieht, zählt jedes solche "Befragen des Orakels" als nur ein Schritt von M).

Wir schreiben $M^O(x)$, um die Ausgabe von M bei Eingabe x mit Orakel O zu bezeichnen.

(b) Nichtdeterministische Orakel-TMen werden analog definiert.

Definition 3.8 (P^O, NP^O)

Für $O \subseteq \{0,1\}^*$ ist

- P^O die Klasse aller Sprachen $L \subseteq \{0,1\}^*$, die durch eine det. Orakel-TM mit Orakel O in polynomiell vielen Schritten entschieden werden können
- NP^O die Klasse aller Sprachen $L \subseteq \{0,1\}^*$, die durch eine ndet. Orakel-TM mit Orakel O in polynomiell vielen Schritten entschieden werden können

Beispiel 3.9

(a) $\overline{SAT} \in P^{SAT}$ durch eine Orakel-TM M, die bei Eingabe einer Formel φ diese auf's Orakelband schreibt, dann in den Zustand q_{query} geht und 0 bzw 1 ausgibt, je nach dem ob M dann in den Zustand q_{yes} bzw q_{no} geht.

(b) Für $O \in P$ ist $P^O = P$, denn:

"$\supseteq$": klar.

"$\subseteq$": Sei M eine det. polyn.zeit Orakel-TM, die eine Sprache $L \in P^O$ entscheidet, und sei N eine det. polyn.zeit TM, die O entscheidet. Jedesmal wenn M das Orakel O befragt, starten wir die TM N. Die dadurch erhaltene TM M' ist dann eine det. polyn.zeit TM, die L entscheidet. ∎

(c) Sei

$$EXPCOM := \{ \langle M, x, 1^t \rangle : M \text{ ist eine det TM, } x \in \{0,1\}^*, t \in \mathbb{N}, \text{ s.d. } M \text{ bei Eingabe } x \text{ nach} \leq 2^t \text{ Schritten den Wert 1 ausgibt} \}$$

Dann gilt:

$$P^{EXPCOM} = NP^{EXPCOM} = EXP \quad \left(\overset{\text{Def}}{=} \bigcup_{c \geq 1} DTIME(2^{(n^c)}) \right)$$

Beweis:

$EXP \subseteq P^{EXPCOM}$:

Sei $L \in EXP$, d.h. es gibt ein $c \geq 1$ und eine det TM M, die L in $2^{(n^c)}$ Schritten entscheidet.

Somit gilt f.a. $x \in \{0,1\}^*$:

$x \in L \Leftrightarrow M$ akzeptiert x nach $\leq 2^{(|x|^c)}$ Schritten

$\Leftrightarrow \langle M, x, 1^t \rangle \in EXPCOM$, für $t = |x|^c$.

Somit ist $L \in P^{EXPCOM}$ durch eine det. polyn.zeit TM M', die bei Eingabe x zunächst den Wert $t = |x|^c$

berechnet, dann das Wort $\langle M, x, 1^{\ell} \rangle$ aufs Orakel-Band schreibt und dann 1 bzw 0 ausgibt je nach dem ob M' nach Befragen des Orakels im Zustand q_{yes} oder q_{no} ist.

$P^{EXPCOM} \subseteq NP^{EXPCOM}$:

Klar.

$NP^{EXPCOM} \subseteq EXP$:

Sei M eine ndet. polynzeit TM, die mit Orakel EXPCOM eine Sprache $L \subseteq \{0,1\}^*$ entscheidet. Sei $c \in \mathbb{N}$ s.d. M bei Eingaben der Länge n stets nach $\leq n^c$ Schritten anhält. Dann gibt es $\leq 2^{(n^c)}$ verschiedene Folgen nichtdeterministischer Entscheidungen, nach denen M verfahren kann. Außerdem kann die Sprache EXPCOM durch eine det. Exponentialzeit-TM entschieden werden. Mittels Durchprobieren aller Folgen ndet. Entscheidungen und Simulation von M mit Berechnung jeder Orakel-Anfrage in Exponentialzeit erhält man eine det. Exponentialzeit-TM, die L entscheidet. Also: $L \in EXP$.

□

Theorem 3.10 (Satz von Baker, Gill und Solovay, 1975)

Es gibt Orakel $A, B \subseteq \{0,1\}^*$ so dass

$$P^A = NP^A \quad \text{und} \quad P^B \neq NP^B$$

Beweis:

Mit $A := EXPCOM$ folgt aus Bsp 2.9 (c), dass $P^A = NP^A$.

Zur Konstruktion von B s.d. $P^B \neq NP^B$ betrachten wir für jedes $B \subseteq \{0,1\}^*$ die unäre Sprache

$$U_B := \{ 1^n : n \in \mathbb{N}, \text{ es } x \in \{0,1\}^n \text{ mit } x \in B \}.$$

Für jedes $B \subseteq \{0,1\}^*$ ist $U_B \in NP^B$, denn bei Eingabe eines Worts der Form 1^n kann eine ndet. Orakel-TM zunächst ein Wort $x \in \{0,1\}^n$ raten und dann das Orakel befragen, ob $x \in B$ liegt.

Wir konstruieren nun ein spezielles $B \subseteq \{0,1\}^*$, s.d. $U_B \notin P^B$ (dann folgt $P^B \neq NP^B$).

Dazu sei für jedes $i \in \mathbb{N}_{>0}$ M_i die det. Orakel-TM, die durch die Binärdarstellung von i (ohne die führende 1) repräsentiert wird.

Wir konstruieren B in Stufen $i = 1, 2, 3, \ldots$, so dass die i-te Stufe bewirkt, dass es eine Zahl $n_i \geq i$ gibt, so dass M_i^B auf Eingaben der Länge n_i die Sprache U_B nicht in $\leq 2^{n_i}/10$ Schritten entscheidet.

Insgesamt folgt daraus dann, dass es keine det. Orakel-TM gibt, die U_B in Polynomialzeit entscheidet.

Konstruktion von B in Stufen $i = 1, 2, 3, \dots$:

Zu Beginn der Konstruktion ist $B = \emptyset$. In jeder Stufe $i \in \mathbb{N}_{>0}$ wird für eine endliche Anzahl von Worten $x \in \{0,1\}^*$ festgelegt, ob $x \in B$ oder $x \notin B$ ist.

Stufe i ($i \in \mathbb{N}_{>0}$):

Sei $n_i \geq i$ groß genug, so dass in den Stufen $j < i$ für kein Wort $x \in \{0,1\}^{n_i}$ festgelegt wurde, ob $x \in B$ oder $x \notin B$ ist.

Lass M_i mit Eingabe 1^{n_i} für $2^{n_i}/10$ Schritte laufen. Wann immer M_i dabei das Orakel für ein Wort $x \in \{0,1\}^*$ befragt, tue folgendes:

- falls in den Stufen $j < i$ bereits festgelegt wurde, ob $x \in B$ oder $x \notin B$ ist, antworte entsprechend mit ja bzw. nein.
- falls in keiner der Stufen $j < i$ festgelegt wurde, ob $x \in B$ oder $x \notin B$ ist, lege fest, dass $x \notin B$ ist

Fall 1: Falls diese Berechnung die Eingabe 1^{n_i} nach $\leq 2^{n_i}/10$ Schritten akzeptiert, so lege fest, dass kein Wort der Länge n_i zu B gehört. Insbes. gilt dann:

$1^{n_i} \notin U_B$.

Falls diese Berechnung die Eingabe 1^{n_i} nicht nach $\leq 2^{n_i}/10$ Schritten akzeptiert, so wähle ein beliebiges Wort $x \in \{0,1\}^{n_i}$ aus, für das bisher noch nicht festgelegt wurde, ob $x \in B$ oder $x \notin B$ ist (ein solches Wort gibt es, da $|\{0,1\}^{n_i}| = 2^{n_i}$ ist, die Berechnung aber höchstens $2^{n_i}/10 < 2^{n_i}$ Orakel-Anfragen macht). Lege fest, dass $x \in B$ ist. Insbes gilt dann:

$1^{n_i} \in U_B$.

In beiden Fällen gilt:

$1^{n_i} \in U_B \iff M_i^B$ akzeptiert die Eingabe 1^{n_i} nicht nach $\leq 2^{n_i}/10$ Schritten. (*)

Angenommen, U_B wird durch eine det. Orakel-TM M in $\leq n^c$ Schritten entschieden (d.h. $U_B \in P^B$).

Wähle dann i hinreichend groß so dass $M = M_i$ und $n_i^c \leq 2^{n_i}/10$.

Aus (*) folgt dann, dass U_B nicht durch M_i^B entschieden wird. ↯ Widerspruch. □

Bemerkung 3.11:

(a) Die Beweise der beiden Zeithierarchiesätze Theorem 3.2 und Theorem 3.5 lassen sich leicht auf Orakel-TMen übertragen (Details: Übung!), so dass man für jedes Orakel $O \subseteq \{0,1\}^*$ erhält:

- $\mathrm{DTIME}^O(n^c) \subsetneq \mathrm{DTIME}^O(n^{c+1})$ (f.a. $c \geq 1$)
- $\mathrm{DTIME}^O(2^{(n^c)}) \subsetneq \mathrm{DTIME}^O(2^{(n^{c+1})})$
- $\mathrm{NTIME}^O(n^c) \subsetneq \mathrm{NTIME}^O(n^{c+1})$ (f.a. $c \geq 2$)
- $\mathrm{NTIME}^O(2^{(n^c)}) \subsetneq \mathrm{NTIME}^O(2^{(n^{c+1})})$
- $P^O \neq EXP^O$, $NP^O \neq NEXP^O$.

(b) Da es Orakel A, B mit $P^A = NP^A$ und $P^B \neq NP^B$ gibt (laut Theorem 3.10), kann

$P \neq NP$ (oder $P = NP$)

daher nicht allein mit Methoden (z.B. Diagonalisierung) bewiesen werden, die sich auch direkt auf Orakel-TMen übertragen lassen.

Kapitel 4: Platzkomplexität

4.1 Platzbeschränkte Berechnungen

Definition 4.1 ((N)SPACE(S(n)))

Sei $S: \mathbb{N} \to \mathbb{N}$. Sei $L \subseteq \{0,1\}^*$

(a) Eine (det. oder ndet.) TM M heißt S(n)-platzbeschränkt, falls sie bei jeder Eingabe $x \in \{0,1\}^*$ höchstens $S(|x|)$ Zellen ihrer Arbeitsbänder benützt.

Bemerke: Jede Zelle darf mehrfach verwendet werden. Die Zellen des (read-only) Eingabebandes werden nicht mitgezählt.

(b) $L \in$ SPACE(S(n)), falls es ein $c \in \mathbb{N}$ und eine det., $c \cdot S(n)$-platzbeschränkte TM gibt, die L entscheidet

(c) $L \in$ NSPACE(S(n)), falls es ein $c \in \mathbb{N}$ und eine $c \cdot S(n)$-platzbeschränkte NDTM gibt, die L entscheidet

Definition 4.2

$S: \mathbb{N} \to \mathbb{N}$ heißt platzkonstruierbar, falls es eine $\mathcal{O}(S(n))$-platzbeschränkte det. TM M gibt, die bei Eingabe von $x \in \{0,1\}^*$ die Zahl $S(|x|)$ berechnet und auf ihrem Ausgabeband ausgibt.

Beispiele 4.3

Die folgenden Funktionen sind platzkonstruierbar:

- $\log n$
- n
- n^c, für jedes $c \in \mathbb{N}$
- 2^n
- $2^{(n^c)}$, für jedes $c \in \mathbb{N}$.

Beweis: Übung.

Fakt 4.4: Für jedes $S: \mathbb{N} \to \mathbb{N}$ gilt:

$$\mathrm{DTIME}(S(n)) \subseteq \mathrm{SPACE}(S(n)) \subseteq \mathrm{NSPACE}(S(n))$$

($\subseteq$ links: da jede $S(n)$-zeitbeschränkte TM nur $\leq S(n)$ Bandzellen besuchen kann; $\subseteq$ rechts: klar)

Konfigurationsgraphen:

Sei M eine (det oder ndet) k-Band TM.

- Eine Konfiguration von M besteht aus
  - dem aktuellen Zustand q von M
  - Zahlen $p_1, \ldots, p_k$, die die aktuellen Kopfpositionen auf den k Bändern angeben.
  - Worten $B_2, \ldots, B_k$ die die aktuellen Beschriftungen der Arbeitsbänder (d.h. der Bänder $2, \ldots, k$) angeben.

≈ Falls M $S(n)$-platzbeschränkt ist, so gilt o.B.d.A.:

- jedes der Worte $B_2, \dots, B_k$ hat Länge $\leq S(n)$, wobei n die Länge des Eingabeworts ist
- jede der Kopfpositionen $p_2, \dots, p_k$ ist $\leq S(n)$
- die Zahl p_1 (Kopfposition auf dem read-only Eingabeband) ist $\leq n+1$.

≈ Die Startkonfiguration C_{start} von M bei Eingabe $*$ ist die Konfiguration mit

- $q = q_{start}$ (Startzustand)
- $p_1 = \dots = p_k = 0$
- $B_2 = \dots = B_k = \triangleright$

≈ Wir gehen davon aus, dass die TM vor dem Anhalten den Inhalt ihrer Arbeitsbänder löscht (abgesehen vom Ausgabeband) und alle Köpfe zurück auf die Startposition setzt.

Somit ist die (einzige) akzeptierende Konfiguration die Konfiguration C_{accept} mit

- $q = q_{halt}$
- $p_1 = \dots = p_k = 0$
- $B_2 = \dots = B_{k-1} = \triangleright$, $B_k = \triangleright 1$

Die (einzige) verwerfende Konfiguration ist die Konfiguration C_{reject} mit

- $q = q_{halt}$
- $p_1 = \dots = p_k = 0$
- $B_2 = \dots = B_{k-1} = \triangleright$, $B_k = \triangleright 0$.

= Der Konfigurationsgraph $G_{M,x}$ von M bei Eingabe x ist der gerichtete Graph, dessen Knoten die möglichen Konfigurationen von M bei Eingabe x sind, und bei dem es eine Kante von einer Konfiguration C zu einer Konfiguration C' gibt, falls M bei Eingabe x in einem Schritt von Konfiguration C zu Konfiguration C' übergehen kann.

Beachte: M akzeptiert x

$\Leftrightarrow$ In $G_{M,x}$ gibt es einen gerichteten Weg von C_{start} zu C_{accept}.

Fakt 4.5:
Sei $S: \mathbb{N} \to \mathbb{N}$, sei M eine $S(n)$-platzbeschränkte (det. oder ndet.) TM, und sei $S(n) \geq \log n$.

(a) Es gibt eine Zahl c (die nur von M's Alphabetgröße, Bänder- und Zustandszahl abhängt), so dass für jede Eingabe $x \in \{0,1\}^*$ gilt:
Jeder Knoten von $G_{M,x}$ kann durch einen Bitstring der Länge $c \cdot S(|x|)$ repräsentiert werden.
Insbes. hat $G_{M,x}$ höchstens $2^{c \cdot S(|x|)}$ Knoten.

(b) Es gibt eine CNF-Formel $\varphi_{M,x}$ der Länge $O(S(|x|)^2)$ über den aussagenlogischen Variablen $y_1, \dots, y_{c \cdot S(|x|)}, y'_1, \dots, y'_{c \cdot S(|x|)}$, so dass für alle Konfigurationen C, C' von $G_{M,x}$ gilt:

Werden in $\varphi_{M,x}$ die Variablen $y_1, \dots, y_{c \cdot S(|x|)}, y'_1, \dots, y'_{c \cdot S(|x|)}$ entsprechend der Bitstrings der Länge $c \cdot S(|x|)$ belegt, die C, C' repräsentieren, so ist

$\varphi_{M,x}$ genau dann erfüllt, wenn es in $G_{M,x}$ eine Kante von C zu C' gibt

Beweis: (a): klar.

(b): Analog zum Beweis des Satzes von Cook und Levin.
Details: Übung! □

Satz 4.6
Sei $S: \mathbb{N} \to \mathbb{N}$ platzkonstruierbar mit $S(n) \geq \log n$. Dann gilt:

$$DTIME(S(n)) \subseteq SPACE(S(n)) \subseteq NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$$

Beweis: Die ersten beiden Inklusionen folgen aus Satz 4.4.
Zur letzten Inklusion: Eine det. TM kann sich bei Eingabe x in Zeit $2^{O(S(|x|))}$ eine Liste sämtlicher Konfigurationen sowie den Konfigurationsgraph $G_{M,x}$ (der gegebenen $S(n)$-platzbeschränkten NDTM M) erzeugen und testen, ob es in $G_{M,x}$ einen Weg von C_{start} zu C_{accept} gibt. Details: Übung! □

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 $n := |V|$.

Klar:
- 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
```

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.

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:

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

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

4.2 PSPACE-Vollständigkeit

Wir wissen bereits, dass $P \subseteq NP \subseteq \mathrm{PSPACE}$ gilt. Da vermutet wird, dass $P \neq NP$ ist, wird auch vermutet, dass $P \neq \mathrm{PSPACE}$ ist — bewiesen kann das bisher aber niemand.

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

Definition 4.12

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

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

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

Bemerkung 4.13

Analog zum Begriff der NP-Vollständigkeit gilt:

(a) Falls es eine PSPACE-vollständige Sprache L' gibt, die in P liegt, so ist P = PSPACE

(b) Die folgende Sprache ist PSPACE-vollständig:

$$\text{SPACE TM SAT} := \{ \langle \alpha, x, 1^s \rangle : \alpha, x \in \{0,1\}^*, s \in \mathbb{N},$$
DTM M_α akzeptiert Eingabe x und nutzt dabei nur $\leq s$ Zellen ihrer Arbeitsbänder $\}$

Beweis: Analog zum Beweis von Theorem 2.12 (NP-Vollständigkeit von TM SAT); Details: Übung!

□