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! □

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 $B \in L$.

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

Beweis:

(a) & (b):

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).

Wir zeigen im Folgenden, dass die Funktion
$h: \{0,1\}^* \to \{0,1\}^*$ mit $h(x) := g(f(x))$ f.a. $x \in \{0,1\}^*$
implizit logspace-berechenbar ist

Beachte: Daraus folgt dann direkt (a), und (b) folgt auch, indem wir annehmen, dass f die Funktion ist, die zeigt, dass $B \leq_e C$ ist und indem wir $g: \{0,1\}^* \to \{0,1\}$ so wählen, dass $g(x) := \begin{cases} 1 & \text{falls } x \in B \\ 0 & \text{sonst} \end{cases}$ (klar: wegen $C \in \mathbb{L}$ ist g implizit logspace-berechenbar). Die Funktion h hat dann die Eigenschaft, dass $h(x) = \begin{cases} 1 & \text{falls } x \in C \\ 0 & \text{sonst} \end{cases}$ — und die implizite logspace-Berechenbarkeit von h zeigt, dass C in L liegt.

Zum Nachweis der impliziten logspace-Berechenbarkeit von h:

(1) h ist polynomiell beschränkt, da $h(x) = g(f(x))$ (f.a. $x \in \{0,1\}^*$) und da f und g polynomiell beschränkt sind

(2) Sei $L_h := \{ \langle x, j \rangle : x \in \{0,1\}^*, j \in \mathbb{N}, h(x)_j = 1 \}$ und
$L'_h := \{ \langle x, j \rangle : x \in \{0,1\}^*, j \in \mathbb{N}, |h(x)| \leq j \}$

Zu zeigen: $L_h \in L$ und $L'_h \in L$

Wir zeigen im Folgenden, dass $L_h \in L$ ist
($L'_h \in L$ kann auf ähnliche Weise gezeigt werden; Details: Übung!)

Wir konstruieren eine DTM M_h, die L_h auf logarithmischer Platz entscheidet. Dabei stellen wir uns vor, dass M_h ein "virtuelles Band" besitzt, auf das sie

bei Eingabe $\langle x,j\rangle$ das Wort $f(x)$ schreibt. Danach wird das virtuelle Band als Eingabeband betrachtet, für das M_g simuliert wird, um zu entscheiden, ob $\langle f(x),j\rangle \in L_g$ liegt (d.h. ob $h(x)_j \overset{\text{Def}}{=} g(f(x))_j = 1$ ist).

Problem: M_h hat nicht genug Platz zur Verfügung, um das gesamte Wort $f(x)$ auf einem Band zu speichern.

Lösung: Stattdessen merkt sich M_h auf einem extra Band die aktuelle Position i, die auf dem virtuellen Band gelesen werden soll und nützt in jedem einzelnen Schritt von M_g die TMen M_f und M'_f, um das auf dem virtuellen Band zu lesende Symbol zu ermitteln: Dieses ist ▷ falls $i=0$; 1 falls $\langle x,i\rangle \in L_f$ (da dann $f(x)_i=1$); □ falls $\langle x,i\rangle \notin L'_f$ (da dann $i > |f(x)|$); und 0 falls $\langle x,i\rangle \in L'_f$ und $\langle x,i\rangle \notin L_f$ (da dann $i \leq |f(x)|$ und $f(x)_i \neq 1$ – also $f(x)_i = 0$).

Platzbedarf von M_h:

- $\sigma(\log(|f(x)|))$ Platz zur Simulation von M_g bei Eingabe $f(x)$

  $\sigma(\log(\text{poly}(|x|))) = \sigma(\log(|x|))$

- $\sigma(\log(|f(x)|))$ Platz zum Speichern der aktuellen Position i auf dem virtuellen Band
- $\sigma(\log(|\langle x,i\rangle|))$ Platz zur Simulation von M_f und M'_f bei Eingabe $\langle x,i\rangle$

  $\sigma(\log(|x|))$

Insgesamt zeigt dies, dass $L_h \in \mathbf{L}$ ist.

(c): analog; Details: Übung! □

Definition 4.24

Sei $C \subseteq \{0,1\}^*$.

(a) C ist NL-hart, falls f.a. $B \in NL$ gilt: $B \leq_e C$.

(b) C ist NL-vollständig, falls $C \in NL$ und C NL-hart ist.

Bemerkung 4.25:

Aus Lemma 4.23 (b) folgt:

Falls es ein NL-vollständiges Problem gibt, das in L liegt, so ist $L = NL$.

Theorem 4.26

Das folgende Problem ist NL-vollständig:

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

Beweis:

PATH $\in NL$ wurde schon in Beispiel 4.8(c) gezeigt.

NL-Härte von PATH:

Sei $B \subseteq \{0,1\}^*$ ein beliebiges Problem in NL und sei M eine NDTM, die B auf Platz $O(\log n)$ entscheidet.

Zu zeigen: $B \leq_e PATH$

Sei dazu $f : \{0,1\}^* \to \{0,1\}^*$ die Funktion mit

$f(x) := \langle G_{M,x}, c_{start}, c_{accept} \rangle$ (f.a. $x \in \{0,1\}^*$).

Klar: $x \in B \iff f(x) \in PATH$.

f ist write-once logspace-berechenbar durch einen Algorithmus, der nacheinander alle Paare C, C' von Konfigurationen von M bei Eingabe x betrachtet (d.h. alle Knoten von $G_{M,x}$) und testet, ob M bei Eingabe x in einem Schritt von Konfiguration C in Konfiguration C' kommt — und entsprechend eine 1 oder eine 00 auf's Ausgabeband schreibt.
Auf diese Weise schreibt der Algorithmus die Kodierung der Adjazenzmatrix von $G_{M,x}$ auf's Ausgabeband. Danach schreibt er noch die Kodierungen von c_{start} und c_{accept} und hält an.

Platzbedarf: In jedem Schritt muss dieser Algorithmus 2 Konfigurationen von M bei Eingabe x speichern — und das geht auf Platz $O(\log|x|)$.

Insgesamt ist f daher write-once logspace-berechenbar und wegen Fakt 4.21 daher auch implizit logspace-berechenbar. Somit ist $B \in_e$ PATH. $\square$

4.4 Nichtdeterministische Platzklassen sind unter Komplementbildung abgeschlossen

Zur Erinnerung:

- Für eine Komplexitätsklasse K ist

$$coK := \{ L \in \{0,1\}^* : \overline{L} \in K \} = \{ \overline{L} : L \in K \}$$

- Vermutung: $NP \neq coNP$

- Bekannt: $NPSPACE \underset{\text{savitch}}{=} PSPACE \underset{\text{deterministisch}}{=} coPSPACE = coNPSPACE$

Theorem 4.27 (Der Satz von Immerman und Szelepcsényi, 1987)

(a) $\overline{PATH} \in NL$

(b) $NL = coNL$

(c) Für jedes platzkonstruierbare $S: \mathbb{N} \to \mathbb{N}$ mit $S(n) \geq \log n$ gilt: $NSPACE(S) = coNSPACE(S)$.

Beweis: (a) Gesucht ist ein nichtdeterministischer Algorithmus B, der auf Platz $O(\log n)$ arbeitet und der bei Eingabe von $\langle G, s, t \rangle$ folgendes Verhalten hat:

(I) falls es in G einen Weg von s nach t gibt, dann soll der Algorithmus bei *jeder* Folge von nichtdet. Entscheidungen ablehnen

(II) falls es in G *keinen* Weg von s nach t gibt, dann soll es mind. eine Folge nichtdet. Entscheidungen geben, bei der der Algo. akzeptiert

Behauptung 1:

Es gibt einen ndet. Algo. A, der auf Platz $O(\log n)$ arbeitet und der bei Eingabe von $\langle G, s \rangle$ folgendes Verhalten hat.

- für jede Folge nichtdeterministischer Entscheidungen gibt A entweder eine Fehlermeldung (kurz: $\perp$) aus, oder die Anzahl r der von s aus in G erreichbaren Knoten
  (d.h. $r = |\{v : v$ ist ein Knoten von G, und es gibt in G einen Weg von s nach $v\}|$)
- es gibt mindestens eine Folge nichtdeterministischer Entscheidungen, so dass A die Zahl r ausgibt.

Bevor wir Beh 1 beweisen, zeigen wir, wie Algo A genutzt werden kann, um den gesuchten Algo B zu erhalten, der zeigt, dass $\overline{PATH} \in NL$ ist.

B geht bei Eingabe $\langle G, s, t \rangle$ wie folgt vor:

1) Lass A mit Eingabe $\langle G, s \rangle$ laufen.
   Falls A eine Fehlermeldung ausgibt, so STOPP mit Ausgabe "nein"
   Sonst sei r die Ausgabe von A (d.h. r ist die Anzahl der von s aus in G erreichbaren Knoten)
2) Zähler := 0; Weg-zu-t-gefunden := FALSE
3) Für jeden Knoten v von G tue folgendes:
   Rate einen in s startenden Weg der Länge $\leq n$ ($= |V(G)|$)
   Falls der Weg mit Knoten v endet, so:
   Zähler := Zähler + 1
   Falls $v = t$, so Weg-zu-t-gefunden := TRUE
4) Falls Zähler $= r$ und Weg-zu-t-gefunden = FALSE, so
   STOPP mit Ausgabe "ja"
   Sonst: STOPP mit Ausgabe "nein".

Man kann sich leicht davon überzeugen, dass B die geforderten Eigenschaften (I) und (II) hat, und dass B auf Platz $O(\log n)$ arbeitet.

Wir müssen also "nur noch" die Behauptung 1 beweisen.

Für jedes $i \leq n = |V(G)|$ sei r_i die Anzahl der von s aus über Wege der Länge $\leq i$ erreichbaren Knoten (insbes: $r_0 = 1$ (nämlich s selbst) und $r_n = r$).

Algo A geht bei Eingabe $\langle G, s \rangle$ wie folgt vor:

```
r_0 := 1 ;  Fehlermeldung := FALSE ;
for i := 1, 2, ..., n do
    Berechne r_i unter Verwendung von r_{i-1}        (1)
Falls Fehlermeldung = TRUE, so
    gib Fehlermeldung aus
sonst gib r := r_n aus.
```

(1) wird dabei wie folgt realisiert:

```
Zähler := 0
Für jeden Knoten v von G do:
    Teste, ob es in G einen Weg der Länge ≤ i von s nach v gibt   (2)
    Falls ja, so Zähler := Zähler + 1
r_i := Zähler
```

② wird dabei wie folgt realisiert.

```
Zähler2 := 0;
Weg-zu-v-gefunden := FALSE
Für jeden Knoten u von G do:
    Teste, ob es in G einen Weg der Länge ≤ i-1 von s nach u gibt (3)
    Falls ja, so
        Zähler2 := Zähler2 + 1
        Falls (u=v oder es in G eine Kante von u zu v gibt), so
            Weg-zu-v-gefunden := TRUE
Falls Zähler2 ≠ r_{i-1}, so
    Fehlermeldung := TRUE
Ausgabe: der Wert der Variablen "Weg-zu-v-gefunden"
```

③ wird dabei wie folgt realisiert:

```
Falls u=s, so
  Stopp mit Ausgabe "ja"
Sonst:
  w := s
  For j := 1, ..., i-1 do
    Rate einen Knoten w' von G
    Falls es keine Kante von w nach w' gibt, so
      Fehlermeldung := TRUE
    Sonst
      w := w'
      Falls w=u, so
        Stopp mit Ausgabe "ja"
```

Per Induktion nach $i = 1, 2, \ldots, n$ lässt sich zeigen:
Ist r_i in korrekt berechnet worden, so wird bei der Berechnung von r_i entweder der korrekte Wert von r_i berechnet oder die Variable "Fehlermeldung" auf TRUE gesetzt, und es gibt mindestens eine Folge von Entscheidungen, die dazu führt, dass bei der "Fehlermeldung" nicht auf TRUE gesetzt wird.

Platzbedarf des Algorithmus:

Speichern von s, i, r_{i-1}, Zähler, v, Zähler2, u, w, j, w' sowie der Booleschen Variablen "Fehlermeldung" und "Weg-zu-v-gefunden".

Insgesamt geht das auf Platz $O(\log n)$.

Damit ist der Beweis von Behauptung 1 fertig, und wir haben gezeigt, dass $\overline{PATH} \in NL$ ist.

□(a)

(b): Zu zeigen: $NL = coNL$

"$\subseteq$":

Sei $B \in NL$ beliebig. Dann ist $B \leq_e PATH$, da PATH NL-vollständig ist (↗ Theorem 4.26).

Somit ist auch $\overline{B} \leq_e \overline{PATH}$.

Wegen $\overline{PATH} \in NL$ (↗(a)) folgt, dass $\overline{B} \in NL$, da NL unter logspace-Reduktionen abgeschlossen ist (↗Lemma 4.23

Somit ist $B \in coNL$. Also: $NL \subseteq coNL$

"$\supseteq$": Sei $C \in coNL$, d.h. $\overline{C} \in NL$. Wegen $NL \subseteq coNL$ ist dann $\overline{C} \in coNL$, also $C = \overline{\overline{C}} \in NL$. Also $coNL \subseteq NL$.

□(b)

(c): Analog zu (a), wobei an Stelle von $\langle G, s, t \rangle$ betrachtet wird, für eine $\langle G_{m,c}, C_{start}, C_{accept} \rangle$ beschränkte NDTM M und eine Eingabe für M.

Details: Übung!

□

4.5 Platzklassen und die Chomsky-Hierarchie

Zur Erinnerung: Die Chomsky-Hierarchie

- Typ 0-Grammatiken: Grammatiken ohne irgendeine Einschränkung
- Typ 1-Grammatiken: kontextsensitive Grammatiken, d.h. Grammatiken, deren Produktionen von der Form $u \to v$ mit $|u| \leq |v|$ sind
- Typ 2-Grammatiken: kontextfreie Grammatiken, d.h. Grammatiken, deren Produktionen von der Form $A \to v$ sind, wobei A ein Nichtterminal ist.
- Typ 3-Grammatiken: reguläre Grammatiken

Aus der Veranstaltung "GL-2: Formale Sprachen und Berechenbarkeit" ist bekannt.

- Typ 0-Grammatiken können genau die rekursiv aufzählbaren (d.h. semi-entscheidbaren) Sprachen erzeugen
- Typ 1-Grammatiken, d.h. kontextsensitive Grammatiken, können genau die Sprachen in NSPACE(n) erzeugen.
  Aus dem Satz von Immerman und Szelepcsényi wissen wir: NSPACE(n) = coNSPACE(n). Daher sind die kontextsensitiven Sprachen unter Komplementbildung abgeschlossen.

- Für Typ 2-Grammatiken, dh kontextfreie Grammatiken, lässt sich das Wortproblem (Eingabe: ein Wort w
Frage: Wird w von der Grammatik erzeugt?)
mittels des CYK-Algorithmus in Zeit $O(n^3)$ lösen. Somit gilt für die Menge KFG aller kontextfreien Sprachen:

$$KFG \subseteq DTIME(n^3) \subsetneq P$$

Außerdem ist bekannt (hier ohne Beweis), dass

$$KFG \subseteq SPACE((\log n)^2)$$

(Lewis, Stearns, Hartmanis, 1965)

Bzgl Typ 3-Grammatiken, dh. reguläre Grammatiken, zeigen wir:

Satz 4.28
Für jedes $S: \mathbb{N} \to \mathbb{N}$ mit $S(n) = o(\log \log n)$ und jedes $L \in SPACE(S)$ gilt: L ist regulär.
Somit ist $SPACE(S) = SPACE(0) = \{L \subseteq \{0,1\}^* : L \text{ regulär}\}$
(kurz: "$SPACE(o(\log\log n)) = REG$")

Beweis:
Sei $S: \mathbb{N} \to \mathbb{N}$ und sei
$L \in SPACE(S)$.
Sei M eine $S(n)$-platzbeschränkte DTM, die L entscheidet
Wegen "linearer Kompression" können wir OBdA annehmen, dass M eine 2-Band DTM ist.

Behauptung 1:
Falls es ein $k \in \mathbb{N}$ gibt mit $S(n) \leq k$ f.a. $n \in \mathbb{N}$, so ist L regulär.

Beweis:
Wegen $S(n) \leq k$ f.a. $n \in \mathbb{N}$, kann der gesamte Inhalt des Arbeitsbandes auch in einer konstanten Anzahl von Zuständen gespeichert werden.
Daher können wir OBdA annehmen, dass M eine 1-Band DTM mit read-only Eingabeband ist.
D.h. M ist ein sog. deterministischer 2-Wege-Automat.

Solche det. 2-Wege-Automaten lassen sich durch herkömmliche NFAs (ndet. endliche Automaten) simulieren. Details: Übung!

▷ Beh. 1.

Sei nun $S(n) = o(\log \log n)$.

Angenommen, L ist nicht regulär.

Wegen Beh. 1 muss es dann für jedes $k \in \mathbb{N}$ ein Wort $x^{(k)} \in \{0,1\}^*$ geben, bei dessen Eingabe M mehr als k Zellen des Arbeitsbandes benutzt.
Sei $x^{(k)}$ ein Wort minimaler Länge, für das dies gilt, und sei $n_k := |x^{(k)}|$ die Länge von $x^{(k)}$.

Klar: $n_1 \leq n_2 \leq n_3 \leq \dots$ und $n_k \xrightarrow[k \to \infty]{} \infty$.

Wir betrachten im Folgenden für jedes $x \in \{0,1\}^*$ die Berechnung $B_M(x)$, die als die Folge von Konfigurationen definiert ist, die M bei Eingabe x durchläuft.

Hier:

Jede Konfiguration C besteht aus

- dem aktuellen Zustand von M
- der aktuellen Kopfposition i auf dem Eingabeband
- dem aktuell auf dem Eingabeband gelesenen Symbol $x_i \in \{0,1,\triangleright,\square\}$
- der aktuellen Kopfposition j auf dem Arbeitsband
- der aktuellen Beschriftung des Arbeitsbandes

Bei einer Eingabe x der Länge n gibt es für jede feste Kopfposition i auf dem Eingabeband höchstens

$$N_n := |Q| \cdot 4 \cdot S(n) \cdot |\Gamma|^{S(n)}$$

verschiedene Konfigurationen, wobei Q und Γ die Zustandsmenge und das Arbeitsalphabet von M sind. Somit gibt es eine Zahl $d \in \mathbb{N}$ so dass

(*) $N_n \leq 2^{d \cdot S(n)}$ f.a. hinreichend großen $n \in \mathbb{N}$ gilt.

Für jede Position i auf dem Eingabeband sei

$$C_{i,M}(x)$$

die Teilfolge der Berechnung $B_M(x)$, die aus allen Konfigurationen besteht, bei denen der Kopf des Eingabebands auf Position i steht.

$C_{i,M}(x)$ wird <u>Crossing-Sequenz für Position i</u> genannt

<u>Klar:</u> In $C_{i,M}(x)$ kommt keine Konfiguration mehrfach vor, denn sonst würde die DTM M bei Eingabe x in eine Endlosschleife kommen — da M aber die Sprache L entscheidet muss M bei jeder Eingabe x irgendwann anhalten.

Somit gilt für die Länge $|C_{i,M}(x)|$ von $C_{i,M}(x)$:

$|C_{i,M}(x)| \leq N_n$, für $n := |x|$

Außerdem gilt für jedes $\ell \in \mathbb{N}$:
Die Anzahl der möglichen Crossing-Sequenzen (für Pos. i) der Länge ℓ ist $\leq N_n^\ell$.

Somit gilt:

(**) Die Gesamtzahl möglicher Crossing-Sequenzen für jedes feste i ist

$$\leq \sum_{\ell=0}^{N_n} N_n^\ell < N_n^{N_n+1} = 2^{(\log N_n)\cdot(N_n+1)} \overset{(*)}{\leq} 2^{d\cdot S(n)\cdot(2^{d\cdot S(n)}+1)}$$

$$\leq 2^{2^{d'\cdot S(n)}}$$ für ein geeignetes $d' \in \mathbb{N}$ und alle hinreichend großen $n \in \mathbb{N}$.

Sei $C'_{i,M}(x)$ die Folge, die aus $C_{i,M}(x)$ entsteht, indem in jeder Konfiguration die Information über die Kopfposition i gelöscht wird.

Behauptung 2:
Sei $x \in \{0,1\}^*$, $n := |x|$ und sei $1 \leq i_1 < i_2 \leq n$ so dass
$C'_{i_1,M}(x) = C'_{i_2,M}(x)$.
Dann gilt für $x' := x_1 \cdots x_{i_1} x_{i_2+1} \cdots x_n$ (d.h.: x' ist das Wort, das aus x entsteht, indem das Teilwort $x_{i_1+1} \cdots x_{i_2}$ gelöscht wird):
$B_M(x')$ ist die Konfigurationenfolge, die aus $B_M(x)$ entsteht, indem

- alle Konfigurationen, bei denen die Kopfposition auf dem Eingabeband eine Position aus $\{i_1+1, \ldots, i_2\}$ ist, gelöscht werden, und
- bei allen Konfigurationen, bei denen die Kopfposition auf dem Eingabeband eine Zahl der Form i_2+j, für $j \geq 1$ ist, diese Zahl ersetzt wird durch die Zahl i_1+j.

Beweis: Übung!

Um den Beweis von Satz 4.28 zu beenden sei nun für jedes $k \in \mathbb{N}$ $x^{(k)} \in \{0,1\}^*$ ein Wort minimaler Länge, bei dessen Eingabe M mehr als k Zellen des Arbeitsbandes benutzt.

D.h. in $B_M(x^{(k)})$ gibt es mindestens eine Konfiguration C, für die die Beschriftung des Arbeitsbandes die Länge $> k$ hat. Sei j die Position des Kopfes auf dem Eingabeband bei C.

Dann gilt:

(1) Die Crossing-Sequenzen $C'_{i,M}(x^{(k)})$ für alle $i \leq j$ müssen alle paarweise verschieden sein, und

(2) die Crossing-Sequenzen $C'_{i,M}(x^{(k)})$ für alle $i \geq j$ müssen alle paarweise verschieden sein.

Denn: Angenommen $C'_{i_1,M}(x^{(k)}) = C'_{i_2,M}(x^{(k)})$ mit $i_1 < i_2 \leq j$ oder $j \leq i_1 < i_2$. Dann gilt gemäß Behauptung 3 für $x' := x_1^{(k)} \cdots x_{i_1}^{(k)} x_{i_2+1}^{(k)} \cdots x_n^{(k)}$, dass $B_M(x')$ die Konfiguration C enthält (bzw. die Variante von C, bei der j ersetzt wurde durch $j - i_2 + i_1$).

D.h. insbesondere, dass M bei Eingabe x' mehr als k Zellen des Arbeitsbandes nutzt. Wegen $|x'| < |x^{(k)}|$ ist dies ein Widerspruch zur Wahl von $x^{(k)}$ als Wort minimaler Länge, bei dessen Eingabe M mehr als k Zellen des Arbeitsbandes nutzt.

Somit muss (1) und (2) gelten.

Insbesondere gibt es daher in $\{ C^{\cdot}_{i,n}(x^{(k)}) : 1 \le i \le |x^{(k)}| = n_k \}$

mindestens $\frac{n_k}{2}$ verschiedene Elemente (für $j \ge \frac{n_k}{2}$ folgt dies aus (1); für $j \le \frac{n_k}{2}$ folgt dies aus (2))

Mit (**) folgt: $\frac{n_k}{2} \le 2^{2^{d' \cdot S(n_k)}}$

Somit: $n_k \le 2 \cdot 2^{2^{d' \cdot S(n_k)}} < 2^{2^{d'' S(n_k)}}$ für ein geeignetes $d'' \in \mathbb{N}$

und daher: $\log\log n_k \le d'' \cdot S(n_k)$,

also $S(n_k) \ge \frac{1}{d''} \cdot \log\log n_k$

↯ Widerspruch zu $S(n) = o(\log\log n)$

Somit muss L regulär sein, und Satz 4.28 ist bewiesen. □

Bemerkung 4.29

Aus Satz 4.28 wissen wir, dass $\mathrm{SPACE}(o(\log\log n))$ genau die regulären Sprachen enthält.

$\mathrm{SPACE}(\log\log n)$ enthält auch nicht-reguläre Sprachen – z.B. die Sprache

$$L_{BIN} := \{ \langle \mathrm{bin}_k(0), \mathrm{bin}_k(1), \cdots \mathrm{bin}_k(2^k - 1) \rangle : k \in \mathbb{N} \}$$

wobei $\mathrm{bin}_k(j)$ die Binärdarstellung der Länge k von j bezeichnet, für j mit $0 \le j < 2^k$.

(Dass L_{BIN} nicht regulär ist, folgt leicht aus dem Pumping-Lemma für reguläre Sprachen. Dass $L_{BIN} \in \mathrm{SPACE}(\log\log n)$ ist, ist eine Übungsaufgabe).

Bemerkung 4.30

Durch Betrachtung von Crossing-Sequenzen kann man für einige konkrete Sprachen untere Schranken für den Platz- oder Zeitbedarf zum Entscheiden dieser Sprachen beweisen.

Z.B. kann man für die Sprache

$$PAL := \{ w \in \{0,1\}^* : w \text{ ist ein Palindrom} \}$$

zeigen:

(1) $PAL \notin SPACE(o(\log n))$

(d.h. es gibt kein $S: \mathbb{N} \to \mathbb{N}$ mit $S(n) = o(\log n)$ so dass $PAL \in SPACE(S(n))$

Zum Vergleich: $PAL \in L = SPACE(\log n)$

(2) PAL wird von keiner 1-Band DTM (mit Schreib-/Lesekopf auf dem Eingabeband) in $o(n^2)$ Schritten entschieden

Zum Vergleich: PAL kann in $O(n^2)$ Schritten von einer 1-Band DTM (mit Schreib-/Lesekopf auf dem Eingabeband) entschieden werden.
Und PAL kann in $O(n)$ Schritten von einer 2-Band DTM entschieden werden.

Details: Übung!

Überblick über die bisher betrachteten Komplexitätsklassen.