11 Hierarchiesätze

Im Folgenden bezeichne $DTime_k(T(n))$ die Klasse aller Sprachen, die von einer $T(n)$-zeitbeschränkten k-Band DTM erkannt werden können. Da die Definition der Klasse $DTime(T(n))$ keine Vorschriften über die (konstante) Anzahl von Bändern macht, ergibt sich $DTime(T(n)) = \cup_{k\geq 1} DTime_k(T(n))$. Die Notationen $DSpace_k(S(n))$, $NTime_k(T(n))$ und $NSpace_k(T(n))$ sind analog zu verstehen. Die uns von früher bekannten Bandreduktionstheoreme, resultierend aus Simulationen von k-Band DTMs auf 1- oder 2-Band DTMs, lesen sich dann wie folgt:

$$\begin{aligned} DSpace(S(n)) &= DSpace_1(S(n)) \\ DTime(T(n)) &\subseteq DTime_1(T(n)^2) \\ DTime(T(n)) &\subseteq DTime_2(T(n)\log T(n)) \end{aligned}$$

Wir werden in diesem Abschnitt zeigen, dass die (asymptotische) Vergrößerung einer Ressourcenschranke zu einer Vergrößerung der Klasse der mit diesen Ressourcen erkennbaren Sprachen führt. Ausagen dieser Art sind unter dem Namen „Hierarchiesätze“ bekannt. Wir beginnen mit folgendem

Lemma 11.1 *Es seien $f_1(n), f_2(n)$ Funktionen mit $f_1(n) = o(f_2(n))$*

1. *Es sei f_2 zeitkonstruierbar. Dann gilt für alle $k \geq 1$:*

$$DTime_k(f_1(n)) \subset DTime_{k+1}(f_2(n)) \ .$$

2. *Es sei $f_2(n) \geq \log n$ platzkonstruierbar. Dann gilt für alle $k \geq 1$:*

$$DSpace_k(f_1(n)) \subset DSpace_{k+1}(f_2(n)) \ .$$

Beweis Die Inklusionsbeziehung ist klar, aber die Echtheit der Inklusion muss jeweils nachgewiesen werden. Der Beweis benutzt die Technik der Diagonalisierung. Wir können eine k-Band DTM M_α mit einer $(k+1)$-Band UTM simulieren, die organisiert ist wie folgt:

- Die ersten k Bänder werden genau so verwendet wie die der k-Band DTM M (bis auf die Tatsache, dass die Symbole aus Γ_M von der UTM binär kodiert werden). Der Kopf von Band 1 führt auf einer Extraspur zudem den Code α mit sich, d.h., bei einer Kopfbewegung wird α um eine Zelle in dieselbe Richtung verschoben.

- Band $k+1$ dient der Kontrolle der Ressourcenschranke $f_2(n)$: Simulationen, welche zur Überschreitung der Ressourcenschranke (oder zu Endlosschleifen) führen würden, werden abgebrochen.[1]

Es ist nicht schwer **zu zeigen**, dass der Zeit- bzw. Platzbedarf der UTM den entsprechenden **Übg.**

[1] Endlosschleifen könnten bei platzbeschränkten Rechnungen auftreten. Da $S_2(n) \geq \log n$ platzkonstruierbar ist, können Endlosschleifen des Simulators aber durch Mitzählen der durchlaufenen Konfigurationen erkannt werden.

Bedarf von M_α maximal um den Faktor $c|\alpha|$ überschreitet (für eine geeignete Konstante c). Wir definieren nun eine Sprache D wie folgt: ein Eingabewort $\alpha 01^p$ wird in D *nicht* aufgenommen gdw die UTM-Simulation zum Akzeptieren von $\alpha 01^p$ durch M_α führt. Umgekehrt ausgedrückt: $\alpha 01^p$ wird in D aufgenommen gdw entweder die UTM-Simulation zum *Nicht-Akzeptieren* von $\alpha 01^p$ durch M_α führt oder die Simulation vorzeitig abgebrochen werden musste. Die Sprache D hat eine geeignete Variante der UTM als Akzeptor. Da per Konstruktion Ressourcenüberschreitungen vermieden werden, gehört D zu $DTime(f_2(n))$ bzw. zu $DSpace(f_2(n))$. Wegen $f_1(n) = o(f_2(n))$ gibt es für jede k-Band DTM M_α mit Ressourcenschranke f_1 einen hinreichend grosen „Polsterparameter“ p, so dass akzeptierende Rechnungen von M_α ohne Resourcenüberschreitung zu Ende simuliert werden können. Offensichtlich (per Diagonalisierung) unterscheidet sich D in mindestens einem Wort von jeder Sprache aus $DTime(f_1(n))$ bzw. aus $DSpace(f_1(n))$. Hieraus ergibt sich die Aussage des Lemmas. **qed.**

Folgerung 11.2 *Es sei $S_2(n)$ platzkonstruierbar und $S_1(n) = o(S_2(n))$. Dann gilt*

$$DSpace(S_1(n)) \subset DSpace(S_2(n)) \ .$$

Beweis Der Beweis ergibt sich unter Ausnutzung von Lemma 11.1 und der Bandreduktionstheoreme aus

$$DSpace(S_1(n)) = DSpace_1(S_1(n)) \subset DSpace_2(S_2(n)) = DSpace(S_2(n)) \ .$$

qed.

Folgerung 11.3 *Es sei $T_2(n)$ zeitkonstruierbar und $T_1(n) \log T_1(n) = o(T_2(n))$. Dann gilt*

$$DTime(T_1(n)) \subset DTime(T_2(n)) \ .$$

Beweis Der Beweis ergibt sich unter Ausnutzung von Lemma 11.1 und der Bandreduktionstheoreme aus

$$DTime(T_1(n)) \subseteq DTime_2(T_1(n) \log T_1(n)) \subset DTime_3(T_2(n)) \subseteq DTime(T_2) \ .$$

qed.

Die in Lemma 11.1 angewendete Diagonalisierungstechnik benutzt drei Grundeigenschaften der beteiligten Komplexitätsklassen:

Universalität: Die Simulation verwendet eine f_2-ressourcenbeschränkte UTM zur Simulation von Maschinen mit Ressourcenschranke f_1.

Kontrollierbarkeit: Ressourcenüberschreitungen werden erkannt und führen zum Abbruch der Simulation.

Abschluss unter Komplement: Die UTM muss die Akzeptanzentscheidung von M_α effizient negieren.

Die ersten beiden Eigenschaften sind auch bei *nichtdeterministischen* Komplexitätsklassen gegeben. Wie sieht es aber mit dem Abschluss unter Komplement aus? Eine Antwort für Platzschranken liefert das folgende Resultat von Immerman und Szelepcsényi:

Satz 11.4 *Für jede platzkonstruierbare Funktion $S(n) \geq \log n$ gilt:*

$$\mathit{NSpace}(S(n)) = \mathit{co\text{-}NSpace}(S(n)) \ .$$

Dieser Satz wird in der Vorlesung „Theoretische Informatik“ für den Spezialfall $S(n) = n$ bewiesen:

- $\mathit{NSpace}(n)$ stimmt überein mit der Klasse der kontextsensitiven Sprachen.
- In der Vorlesung „Theoretische Informatik“ wird gezeigt, dass kontextsensitive Sprachen abgeschlossen sind unter Komplement.

Da der Beweis für den allgemeinen Fall nicht wesentlich aufwändiger ist als für den Spezialfall $S(n) = n$, wollen wir ihn im Rahmen der Vorlesung „Komplexitätstheorie“ nicht im Detail führen, sondern in der folgenden Beweisskizze nur die wesentlichen Ideen ins Gedächtnis rufen:

Beweisskizze

- Für eine Sprache $L \in \mathit{NSpace}(S(n))$ ist zu zeigen $\bar{L} \in \mathit{NSpace}(S(n))$. Es sei M eine $S(n)$-platzbeschränkte NTM, die L erkennt. Es sei $G_M(w)$ der Konfigurationsdigraph von M bei Eingabe w, dessen Knoten die Konfigurationen von M repräsentieren und bei welchem die Startkonfiguration $K_0(w)$ als eine Art „Startknoten“ ausgezeichnet wird. Wir haben eine Kante von K nach K' gdw K' eine direkte Folgekonfiguration von K ist. Wir dürfen oBdA annehmen, dass es eine eindeutige akzeptierende Endkonfiguration K_+ gibt. Eine NTM $\bar{M}$ für die Sprache $\bar{L}$ muss für jedes Wort $w \in \bar{L}$ nichtdetermistisch verifizieren können, dass in $G_M(w)$ *kein* Pfad von $K_0(w)$ nach K_+ existiert. Ein Pfad ließe sich durch Raten desselben verifizieren. Das Rätsel lautet: wie lässt sich die Nicht-Existenz eines Pfades verifizieren?

- Angenommen wir wüssten, wieviele Knoten in $G_M(w)$ von $K_0(w)$ aus erreichbar sind, sagen wir $N = 2^{O(S(n))}$ viele. Dann können wir die Nicht-Erreichbarkeit von K_+ dadurch verifizieren, dass wir zu N von K_+ verschiedenen erreichbaren Knoten K jeweils einen Pfad von $K_0(w)$ nach K raten.

- Die Anzahl N der von $K_0(w)$ aus erreichbaren Knoten lässt sich mit der Technik des (nichtdeterministischen) induktiven Zählens bestimmen. Dabei wird Folgendes erreicht:
  - Entweder $\bar{M}$ bestimmt N korrekt oder $\bar{M}$ bricht den Zählprozess erfolglos ab.
  - Es existiert mindestens eine Rechnung von $\bar{M}$, die N korrekt bestimmt.

Weitere Details zum induktiven Zählen können im Skriptum zur Vorlesung „Theoretische Informatik“ nachgelesen werden. Beachten Sie, dass ein Binärzähler, der bis $N = 2^{O(S(n))}$ zählen kann, nur $O(S(n))$ Zellen beansprucht. Bei geschickter Implementierung von $\bar{M}$ ergibt sich die Platzschranke $S(n)$, womit dann $\bar{L} \in \mathit{NSpace}(S(n))$ bewiesen wäre. **qed.**

Da für nichtdeterministische Platzkomplexitätsklassen alle drei Voraussetzungen — Universalität, Kontrollierbarkeit, Abschluss unter Komplement — vorliegen, ergibt sich mit Beweisen analog zu denen für den deterministischen Fall:

Folgerung 11.5 *Es sei $S_2(n)$ platzkonstruierbar und $S_1(n) = o(S_2(n))$. Dann gilt*

$$\mathit{NSpace}(S_1(n)) \subset \mathit{NSpace}(S_2(n)) \ .$$

Für nichtdeterministische Zeitkomplexitätsklassen ist der Abschluss unter Komplement vermutlich nicht gegeben. (Zum Beispiel wird $NP \neq$ co-NP vermutet.) Erstaunlicherweise lässt sich dennoch der folgende Zeithierarchiesatz beweisen:

Satz 11.6 *Es sei T_2 zeitkonstruierbar und $T_1(n) \leq T_1(n+1) = o(T_2(n))$. Dann gilt für alle $k \geq 1$: $\mathit{NTime}_k(T_1) \subset \mathit{NTime}_{k+1}(T_2)$.*

Beweis Der Beweis benutzt die Technik der „lazy diagonalization” zur Konstruktion einer Sprache $D \in \mathit{NTime}_{k+1}(T_2(n)) \setminus \mathit{NTime}_k(T_1(n))$. Viele technische Details sind ähnlich wie im Beweis von Lemma 11.1, so dass wir uns hier auf die zusätzlichen Manöver konzentrieren, die der fehlende Abschluss unter Komplement uns aufnötigt. Es sei $f : \mathbb{N} \to \mathbb{N}$ die folgende induktiv definierte Funktion:

$$f(1) = 2 \ \text{ und } \ f(i+1) = 2^{T_2(f(i))}$$

Es ist nicht schwer zu zeigen, dass sich zu jedem $n \in \mathbb{N}$ der kleinste Index i mit

$$f(i) < n \leq f(i+1)$$

von einer $T_2(n)$-zeitbeschränkten DTM berechnen lässt. Wir erinnern daran, dass wir mit $\alpha(i)$ den i-ten Binärstring in der natürlichen Aufzählung aller Binärstrings bezeichnen. Im Rahmen der „lazy diagonalization“ soll für jede $T_1(n)$-zeitbeschränkte NTM M Folgendes erreicht werden: wenn $\alpha(i) = \alpha 01^p$ ein Codewort für M mit hinreichend großer Polsterung p ist, dann soll D sich von $L(M)$ auf einem Wort der Form 1^n mit $f(i) < n \leq f(i+1)$ unterscheiden. Hierzu gehen wir bei gegebener Eingabe 1^n vor wie folgt:

Fall 1: $f(i) < n < f(i+1)$.
Dann simuliere $M = M_{\alpha(i)}$ nichtdeterministisch auf Eingabe 1^{n+1} und nimm 1^n in D auf gdw die simulierte Rechnung zum Akzeptieren führt (*keine* Abänderung des Akzeptanzverhaltens). Beachte, dass die Simulation wegen der Voraussetzung $T_1(n+1) = o(T_2(n))$ im Rahmen der Zeitschranke $T_2(n)$ geleistet werden kann.

Fall 2: $n = f(i+1)$.

Dann simuliere $M = M_{\alpha(i)}$ *deterministisch* auf Eingabe $1^{f(i)+1}$ und nimm 1^n in D auf gdw die simulierte Rechnung *nicht* alzeptiert. Da M auf Eingaben der Länge $f(i)+1$ maximal $cT_1(f(i)+1)$ Schritte rechnet (für eine geeignet gewählte Konstante c), gibt es im Konfigurationsdigraphen von M maximal $2^{cT_1(f(i)+1)} = 2^{o(T_2(f(i)))}$ Berechnungspfade und die deterministische Simulation kann von einer $2^{o(T_2(f(i)))}$-zeitbeschränkten DTM erledigt werden. Wegen $f(i+1) = 2^{T_2(f(i)}$ ist dann erst recht die Zeitschranke $T_2(f(i+1)) = T_2(n)$ ausreichend.[2]

Wir argmentieren nun, dass D sich von M (hinreichend große Polsterung des Codeworts von M unterstellt) in mindestens einem der Strings 1^n, $f(i) < n \leq f(i+1)$, unterscheidet. Beachte, dass, gemäß der obigen Fallunterscheidung,

$$\forall n = f(i+1), \ldots, f(i+1) - 1 : D(1^n) = M(1^{n+1}) \text{ und } D(1^{f(i+1)}) \neq M(1^{f(i)+1}) \ . \quad (1)$$

Wir machen die (heuchlerische) Annahme, dass

$$\forall n = f(i)+1, \ldots, f(i+1) : D(1^n) = M(1^n) \ . \quad (2)$$

Bedingung (1) kombiniert mit (2) führt dann aber zu einem Widerspruch. Daher ist die Annahme (2) falsch und D unterscheidet sich für mindestens einen Index n auf 1^n von M — wie gewünscht. **qed.**

Folgerung 11.7 *Es sei $T_1(n)$ und $T_2(n)$ zeitkonstruierbar mit $T_1(n) \leq T_1(n+1) = o(T_2(n))$. Dann gilt: $NTime(T_1) \subset NTime(T_2)$.*

Beweis 2-Band NTMs sind genau so zeiteffizient wie k-Band NTMs. Es gilt nämlich für jede zeitkonstruierbare Funktion $T(n)$ folgendes Bandreduktionstheorem: **Übg.**

$$NTime(T(n)) = NTime_2(T(n))$$

In Kombination mit Satz 11.6 ergibt sich

$$NTime(T_1(n)) = NTime_2(T_1(n)) \subset NTime_3(T_2(n)) = NTime(T_2(n)) \ ,$$

was den Beweis abschließt. **qed.**

Offenes Problem: Wie erinnern an die Platz-Zeit-Hierarchie

$$\mathcal{L} \subseteq \mathcal{NL} \subseteq P \subseteq NP \subseteq PSpace \quad (3)$$

Aus dem nichtdeterministischen Platzhierarchiesatz folgt $\mathcal{NL} \subset PSpace$. Demnach muss die Inklusionskette (3) an irgendeiner Stelle zerreißen, d.h., mindestens eine der Inklusionen muss echt sein. Es wird vermutet, dass *alle* Inklusionen in (3) echt sind. Aber von keiner einzelnen konnte dies bis dato bewiesen werden.

[2]Da in beiden diskutierten Fällen Asymptotik im Spiel ist, stimmt die vorgetragene Argumentation, dass Zeitschranke $T_2(n)$ für die Simulation ausreicht, streng genommen erst bei hinreichend großer Polsterung p (was das Diagonalisierungsargument aber nicht beschädigt).

12 Die Klasse NL

Ein Verwandter des P–NP Problems ist das $\mathcal{L}$–$\mathcal{NL}$ Problem: Ist die Inklusion $\mathcal{L} \subseteq \mathcal{NL}$ echt? Da dieses Problem trotz großer Anstrengungen bis heute nicht gelöst werden konnte, ist es naheliegend, nach „schwersten Problemen“ in $\mathcal{NL}$ zu suchen. Diese können, in Analogie zur NP-Vollständigkeitstheorie, wieder mit Hilfe von Problemreduktionen dingfest gemacht werden. Allerdings ist es nicht zweckmäßig, Karp-Reduktionen zu verwenden: aus $L_1 \leq_{pol} L_2$ und $L_2 \in \mathcal{L}$ folgt nämlich *nicht*, dass $L_1 \in \mathcal{L}$, da die zugrunde liegende Reduktionsabbildung zwar in Polynomialzeit, aber nicht notwendig mit logarithmischem Platz, berechnet werden kann. Die für die Berechnung der Reduktionsabbildung zur Verfügung gestellten Ressourcen sollten sich stets an der kleineren der beiden beteiligten Komplexitätsklassen orientieren, in unserem Fall also an de Klass $\mathcal{L}$. Dies führt uns zum Begriff der logspace-Reduktionen:

Definition 12.1 *Eine Sprache L_1 heißt* logspace-reduzierbar *auf L_2, notiert als $L_1 \leq_{log} L_2$, falls eine Abbildung $f : \Sigma^* \to \Sigma^*$ existiert mit:*

1. *f ist* logspace-berechenbar*, d.h., f ist von einer $log(n)$-platzbeschränkten DTM, versehen mit einem Read-only Eingabeband, einem Write-only Ausgabeband sowie einem Arbeitsband, berechenbar. Dabei wird nur der Platzverbrauch auf dem Arbeitsband gezählt.*

2. *Für alle $x \in \Sigma^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$.*

Da k-Band DTMs ohne Verlust an Platzeffizienz von 1-Band DTMs simuliert werden können, ändert sich der Begriff der logspace-Berechenbarkeit von f nicht, wenn wir der betreffenden DTM k Arbeitsbänder (statt nur einem) spendieren.

Die Relation „$\leq_{log}$” besitzt die gewünschten Eigenschaften:

Lemma 12.2 1. *Aus $L_1 \leq_{log} L_2$ und $L_2 \leq_{log} L_3$ folgt $L_1 \leq_{log} L_3$.*

2. *Aus $L_1 \leq_{log} L_2$ und $L_2 \in \mathcal{L}$ folgt $L_1 \in \mathcal{L}$.*

Beweis

1. Es sei f die Reduktionsabbildung zu $L_1 \leq_{log} L_2$ und g die Reduktionsabbildung zu $L_2 \leq_{log} L_3$. Wir möchten zeigen, dass $g \circ f$ eine Reduktionsabbildung zu $L_1 \leq_{log} L_3$ ist. Freilich gilt für alle $x \in \Sigma^*$

$$x \in L_1 \Leftrightarrow f(x) \in L_2 \Leftrightarrow g(f(x)) \in L_3 \ .$$

   Aber ist $g \circ f$ logspace-berechenbar? Da $y = f(x)$ einerseits die Ausgabe der f-Berechnung, andererseits aber die Eingabe der g-Berechnung ist, scheint es folgendes Dilemma zu geben:

   - Wenn die Ausgabe y zur ersten Reduktion auf das Write-only Ausgabeband geschrieben wird, dann darf die Prozedur, die $g(y) = g(f(x))$ berechnen soll ihre Eingabe y nicht lesen.

- Logarithmischer Speicher reicht aber nicht aus, um $f(x)$ in Gänze auf das Arbeitsand zu schreiben. Schlimm, schlimm, ...

Der Ausweg aus dem Dilemma ist wie folgt. Wir halten eine Simulation der DTM M_g zur Berechnung von $g(f(x))$ aufrecht und merken uns in einem Zähler die aktuelle Position des Read-only Eingabekopfes von M_g (anfangs die Position 1). Wenn M_g das j-te Symbol der (nicht vorhandenen!) Eingabe $f(x)$ lesen möchte, starten wir die DTM M_f zur Berechnung von $f(x)$, unterdrücken die Ausgabe der ersten $j-1$ Bits und benutzen das j-te Ausgabebit, um den nächsten Berechnungsschritt von M_g zu simulieren. Die Details dieser Simulation sind leicht auszuarbeiten.

2. Es sei f die Reduktionsabbildung zu $L_1 \leq_{log} L_2$. L_1 kann im Rahmen einer logspace-Berechnung erkannt werden, indem wir die DTM zur Berechnung von $f(x)$ und die DTM, welche „$f(x) \in L_2$?“ entscheidet, kooperieren lassen. Das Eingabe/Ausgabe-Dilemma wird wieder mit dem soeben geschilderten Trick gelöst.

qed.

Dieser Beweis macht etwas deutlich, was wir schon häufiger beobachtet haben: ein Markenzeichen von „logspace“ ist, dass der zur Verfügung stehende Platz (gerade soeben) ausreicht, um einen (oder konstant viele) Zähler zu speicher, die von 0 bis poly(n) zählen können.

Gegeben unsere Erfahrung mit der NP-Vollständigkeitstheorie ist die folgende Definition und das folgende Lemma „Standard“:

Definition 12.3 *Eine Sprache L_0 heißt $\mathcal{NL}$-*hart*, falls $L \leq_{log} L_0$ für alle $L \in \mathcal{NL}$. Falls darüber hinaus $L_0 \in \mathcal{NL}$, dann heißt L_0 $\mathcal{NL}$-vollständig.*

Lemma 12.4 *Es gelte $L_1 \leq_{log} L_2$ und L_1 sei $\mathcal{NL}$-hart. Dann ist auch Ł$_2$ $\mathcal{NL}$-hart.*

Ausgehend von einem $\mathcal{NL}$-harten Problem können wir mit Hilfe von logspace-Reduktionen nach und nach einen Stammbaum $\mathcal{NL}$-harter Probleme „wachsen lassen“. Wie gelangen wir an ein erstes $\mathcal{NL}$-hartes Problem? Voilà, hier ist es:

Digraph Reachability: Gegeben ein Graph $G = (V, E)$ mit zwei ausgezeichneten Koten $s, t \in V$, gibt es einen Pfad in G von s nach t?

Satz 12.5 *„Digraph Reachability“ ist $\mathcal{NL}$-vollständig.*

Beweis Die Mitgliedschaft von „Digraph Reachability“ in der Klasse $\mathcal{NL}$ ergibt sich wie folgt: rate einen Pfad von s nach t und durchlaufe ihn dabei, speichere aber auf dem Arbeitsband immer nur den Knoten, an dem der Pfaddurchlauf sich aktuell befindet.
Es sei $L \in \mathcal{NL}$ beliebig aber fest. Weiter sei M eine $log(n)$-platzbeschränkte NTM, die L erkennt. Zu einer Eingabe w sei $K_M(w)$ der resultierende Konfigurationsdigraph mit ausgezeichneten Knoten (=Konfigurationen) $K_0(w)$ (Startkonfiguration) und K_+ (akzeptierende Endkonfiguration). Die Abbildung

$$w \mapsto G_M(w), K_0(w), K_+$$

hat die Eigenschaft

$$w \in L \Leftrightarrow \text{es gibt in } G_M(w) \text{ einen Pfad von } K_0(w) \text{ nach } K_+ \ ,$$

und es ist nicht schwer zu sehen, dass sie logspace-berechenbar iat.
Unsere Diskussion hat ergeben, dass „Digraph Reachability“ $\mathcal{NL}$-vollständig ist. **qed.**

Da logspace-Rechnungen in Polynomialzeit durchgeführt werden können, gilt: $L_1 \leq_{log} L_2 \Rightarrow L_1 \leq_{pol} L_2$. Wir merken kurz an, dass alle von uns bisher vollzogenen Karp-Reduktionen mit etwas Liebe und Sorgfalt in logspace-Reduktionen transformiert werden können. Insbesondere sinf SAT, 3-SAT sowie alle weiteren Sprachen des von uns bisher entwickelten Stammbaums NP-vollständig unter logspace-Reduktionen. Aus der Existenz eines $\log(n)$-platzbeschränkten Algorithmus für SAT (oder 3-SAT, ...) würde also $\mathcal{L} = NP$ folgen!

Sprachen aus NP sind dadurch charakterisiert, dass es für jedes Wort $x \in L$ einen kurzen und zeiteffizient verifizierbaren *Beweis* (manchmal auch *Zertifikat* genannt) y gibt, der die Mitgliedschaft von x in L belegt. Sprachen aus $\mathcal{NL}$ sind in ähnlicher Weise durch kurze und platzeffizient verifizierbare *Read-once Zertifikate* charakterisiert:

Definition 12.6 *Ein* Read-once Eingabeband *ist ein Read-only Eingabeband, auf welchem der Kopf in jedem Schritt eine Position nach rechts rückt. Für eine DTM M mit einem Read-only Eingabeband und einem zusätzlichen Read-once Eingabeband bezeichnet $M(x, y) \in \{0, 1\}$ die Ausgabe von M zu Eingabe x auf dem Read-only Eingabeband und Eingabe y auf dem Read-once Eingabeband („1“ = Akzeptieren).*

Satz 12.7 *Eine Sprache L ist aus $\mathcal{NL}$ gdw ein Polynom $p(n)$ und eine (auf dem Arbeitsband) $\log(n)$-platzbeschränkte DTM M existieren, so dass folgendes gilt:*

1. *M ist neben ihrem Read-only Eingabeband, ihrem Write-only Ausgabeband und ihrem Arbeitsband mit einem weiteren Read-once Eingabeband ausgestattet.*
2. *Für alle $x \in \Sigma^*$: $x \in L \Leftrightarrow \exists y \in \{0, 1\}^{p(|x|)} : M(x, y) = 1$.*

In Verbindung mit Satz 12.7 heißt y auch das „Read-once Zertifikat“ für $x \in L$. Der (offensichtliche) Beweis dieses Satzes benutzt das Rate-Verifikationsprinzip, welches wir im Zusammenhang mit der Klasse NP bereits kennen gelernt haben.

Aus dem Satz von Immerman und Szelepsènyi ergibt sich die

Folgerung 12.8 *$\mathcal{NL} = co\text{-}\mathcal{NL}$.*

Da $\mathcal{NL}$ abgeschlossen unter Komplement ist gehört „Digraph Unreachability“ (das Komplement von „Digraph Reachability”) ebenfalls zu $\mathcal{NL}$. In Verbindung mit Satz 12.7 heißt das, dass die *Nicht-Existenz* eines Pfades von s nach t in einem Digraphen G sich durch ein kurzes und logspace-verifizierbares Read-once Zertifikat belegen lässt.

Wir stellen abschließend die Frage, ob die in Satz 12.7 beschriebene DTM mächtiger wird, wenn das Zertifikat y auf ein zusätzliches Read-only Eingabeband geschrieben worden wäre (anstatt auf ein Read-once Eingabeband). Die Antwort, vielleicht etwas überraschend, lautet „viel mächtiger“: mit DTMs von diesem Zuschnitt lassen sich exakt die Sprachen der Klasse NP erkennen. Den Beweis dieser Aussage empfehlen wir als **Übung**. **Übg.**

13 PSpace-vollständige Probleme

13.1 Quantifizierte Boolesche Formeln

Definition 13.1 *Die Sprache der* Wahren Quantifizierten Booleschen Formeln*, notiert als TQBF, enthalte alle (Kodierungen von) quantifizierten Booleschen Formeln der Form*

$$(\mathcal{Q}_1 v_1) \dots (\mathcal{Q}_m v_m) F(v_1, \dots, v_m) \ , \tag{4}$$

welche die Korrektheitsbedingung

$$\mathcal{Q}_1 a_1 \in \{0,1\}, \dots, \mathcal{Q}_m a_m \in \{0,1\} : F(a_1, \dots, a_m) \tag{5}$$

erfüllen. Hierbei sei m eine beliebige nicht-negative ganze Zahl, $\mathcal{Q}_1, \dots, \mathcal{Q}_m \in \{\exists, \forall\}$, und F sei eine Boolesche Formel in den Booleschen Variablen $v_1, \dots, v_m$.

Beispiel 13.2 *Folgende quantifizierten Booleschen Formeln haben die in (4) vorgeschriebene Form:*

$$\begin{array}{lcll} qbf_1 & := & (\forall v_1)(\exists v_2)(\exists v_3) & \bar{v}_3 \wedge ((\bar{v}_1 \wedge v_2) \vee (v_1 \wedge \bar{v}_2)) \\ qbf_2 & := & (\exists v_1)(\forall v_2)(\exists v_3) & \bar{v}_3 \wedge ((\bar{v}_1 \wedge v_2) \vee (v_1 \wedge \bar{v}_2)) \end{array}$$

Wir werden weiter unten sehen, dass qbf_1 die Bedingung (5) erfüllt und somit zu TQBF gehört, wohingegen qbf_2 diese Bedingung verletzt und somit nicht zu TQBF gehört.

Satz 13.3 *TQBF $\in$ PSpace.*

Beweis Wir verwenden zunächst eine platzineffiziente EXHAUSTIVE SEARCH, um ein kanonisches Auswertungsschema klarzumachen. Anschließend verwenden wir die Technik des *Ariadne Fadens*[3], um die EXHAUSTIVE SEARCH platzeffizient zu gestalten.
Wir gehen aus von iener Eingabe der Form (4). Mit einem vollständig binären Entscheidungsbaum T der Tiefe m lassen sich alle 2^m denkbaren Belegungen von $v_1, \dots, v_m$ durchprobieren. Wir markieren die inneren Knoten von T der Tiefe $i-1$ mit $\mathcal{Q}_i \in \{\exists, \forall\}$, um deutlich zu machen, wie die Variable v_i quantifiziert ist.

Einschub Die Entscheidungsbäume T_1 und T_2, die zu den Formeln qbf_1 und qbf_2 aus Beispiel 13.2 korrespondieren, sind in den Abbildungen 1 und 2 zu besichtigen. Der Fortgang des Beweises kann an diesen Abbildungen nachvollzogen werden.

Jedes Blatt von T korrespondiert zu einer vollständigen Belegung $(b_1, \dots, b_m) \in \{0,1\}^m$ der Variablen $(v_1, \dots, v_m)$. Ein innerer Knoten u der Tiefe i in T korrespondiert zu einer partiellen Belegung von $(v_1, \dots, v_i)$ mit $(b_1, \dots, b_i)$. Unser Ziel ist, die Knoten von T „bottom-up“ mit den Booleschen Wahrheitswerten 0 oder 1 zu markieren, wobei ein zur partiellen

[3]der Sage nach erstmals von Ariadne angewendet, um Theseus davor zu schützen, sich im Labyrinth des Minotaurus zu verirren

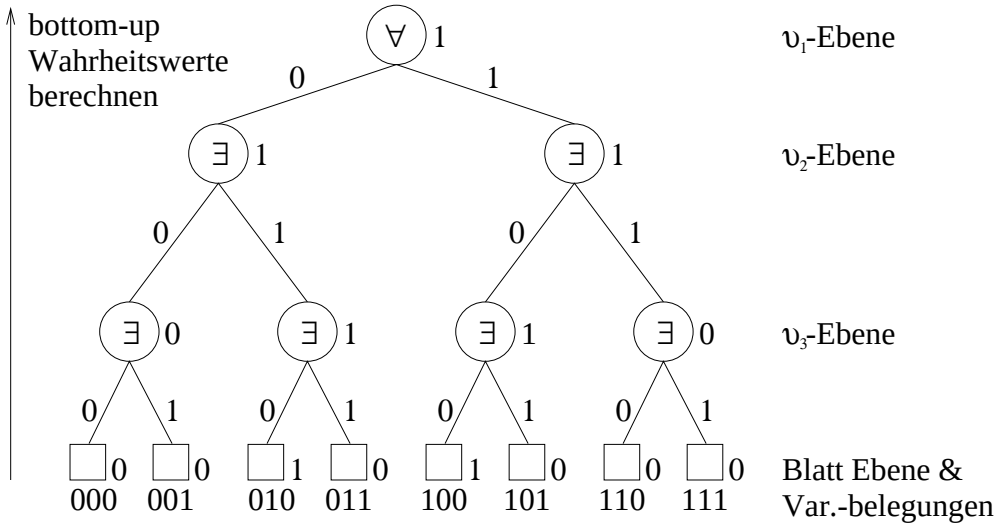

Abbildung 1: Der Entscheidungsbaum T_1 für die quantifizierte Boolesche Formeln qbf_1 aus Beispiel 13.2.

Belegung $(b_1, \ldots, b_i)$ korrespondierender Knoten u der Tiefe i genau dann mit 1 markiert werden soll, wenn die Bedingung

$$\mathcal{Q}_{i+1}a_{i+1} \in \{0,1\} \cdots \mathcal{Q}_m a_m \in \{0,1\} : F(b_1, \ldots, b_i, a_{i+1}, \ldots, a_m) \tag{6}$$

erfüllt ist. Zur Markierung eines zur vollständigen Belegung $b = (b_1, \ldots, b_m)$ korrespondierenden Blattes, brauchen wir lediglich die Formel F an b auszuwerten. Wenn wir induktiv annehmen, dass die Kinder u_0, u_1 eines inneren Knoten u bereits korrekt mit Wahrheitswerten w_0, w_1 markiert sind, dann führt folgende Regel zu einer korrekten Markierung von u:

- Wenn u mit Quantor $\exists$ markiert ist, dann markiere u mit Wahrheitswert $w_0 \vee w_1$.
- Wenn u mit Quantor $\forall$ markiert ist, dann markiere u mit Wahrheitswert $w_0 \wedge w_1$.

Auf diese Weise können wir, beginnend bei den Blättern, alle Knoten von T bottom-up mit Wahrheitswerten markieren bis wir schließlich bei der Wurzel anlangen. Aus der Invarianzbedingung (6) geht durch Inspektion des Spezialfalles $i = 0$ hervor, dass die Wurzel von T genau dann mit 1 markiert ist, wenn die gesamte quantifizierte Formel zu TQBF gehört. Wir akzeptieren also die Eingabe genau dann, wenn am Ende unserer Markierungsprozedur die Wurzel des Entscheidungsbaumes T mit Wahrheitswert 1 markiert ist.
Entscheidungsbaum T hat 2^m Blätter. Seine Größe ist i.A. nicht polynomiell in der Eingabelänge n beschränkt. Eine platzeffiziente EXHAUSTIVE SEARCH darf zu keinem Zeitpunkt den Baum T vollständig abspeichern. Stattdessen verwenden wir beim Durchlaufen von T die Technik des „Ariadne-Fadens“.
Um die eingangs beschriebene Markierungsprozedur platzeffizient durchzuführen, wird T in

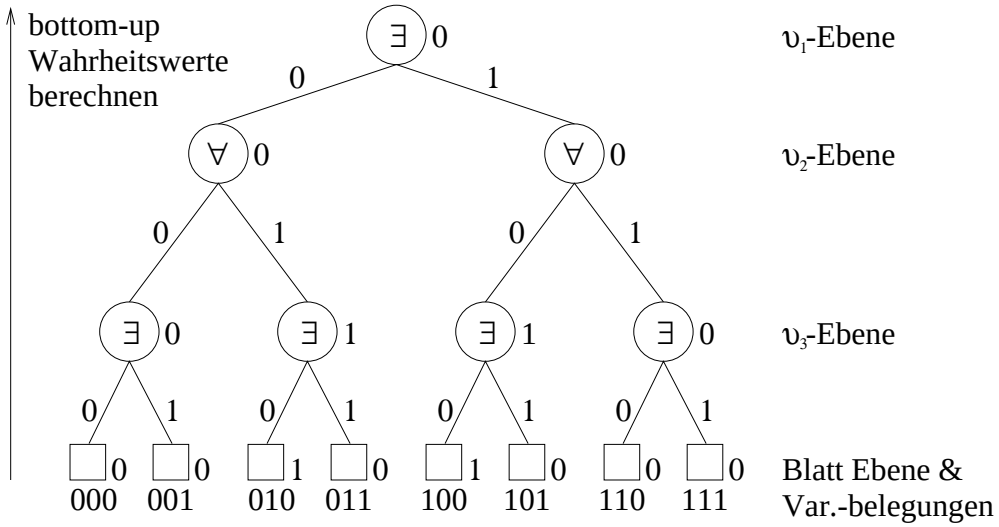

Abbildung 2: Der Entscheidungsbaum T_2 für die quantifizierte Boolesche Formel qbf_2 aus Beispiel 13.2.

Präordnung durchlaufen, wobei wir zu jedem Zeitpunkt nur den Pfad von der Wurzel zum aktuellen Knoten (Ariadne-Faden) und die aktuelle partielle Belegung $(b_1, \ldots, b_i)$ in kellerartig[4] abspeichern. Zusätzlich speichern wir zu jedem Knoten auf dem Ariadne-Faden einen Record der folgenden Form:

$\forall$	w_0	?	?
Quantor	Wahrheitswert linker Sohn	Wahrheitswert rechter Sohn	Wahrheitswert aktueller Knoten

Hieraus kann man ersehen, dass pro Knoten auf dem Ariadne Faden nur konstanter Platz erforderlich ist, und dass die Information in den Records ausreicht, um die oben beschriebene Markierungsprozedur am Laufen zu halten. Da in einem Blatt des Entscheidungsbaumes die Boolesche Formel F auf gegebenen Bits $b_1, \ldots, b_n$ auszuwerten ist, genügt Speicherplatz der Grösenordnung $n + |F|$, wobei $|F|$ die Kodierungslänge von F bezeichnet. Aus unserer Diskussion geht hervor, dass TQBF zur Klasse *PSpace* gehört. **qed.**

Satz 13.4 *TQBF ist PSpace-hart*

Beweis Sei $L \in PSpace$. Wir haben zu zeigen, dass $L \leq_{pol}$ TQBF. Sei M eine DTM, die L erkennt und, für ein Polynom S, auf Eingaben w der Länge n nur $S(n)$ Zellen besucht. Dann hat die Rechnung von M auf w für eine geeignete Konstante c maximal $2^{cS(n)}$ Konfigurationen. Offensichtlich ist $w \in L$ äquivalent dazu, dass die Startkonfiguration $K_0(w)$

[4]Entweder der Ariadne Faden wird um 1 länger (wenn wir tiefer in den Entscheidungsbaum absteigen) oder um 1 kürzer (wenn wir wieder hochsteigen). Dies entspricht den Operationen des Schreibens und Löschens auf dem Kellerspeicher. Eine DTM würde eines ihrer Bänder als Keller bereit stellen.

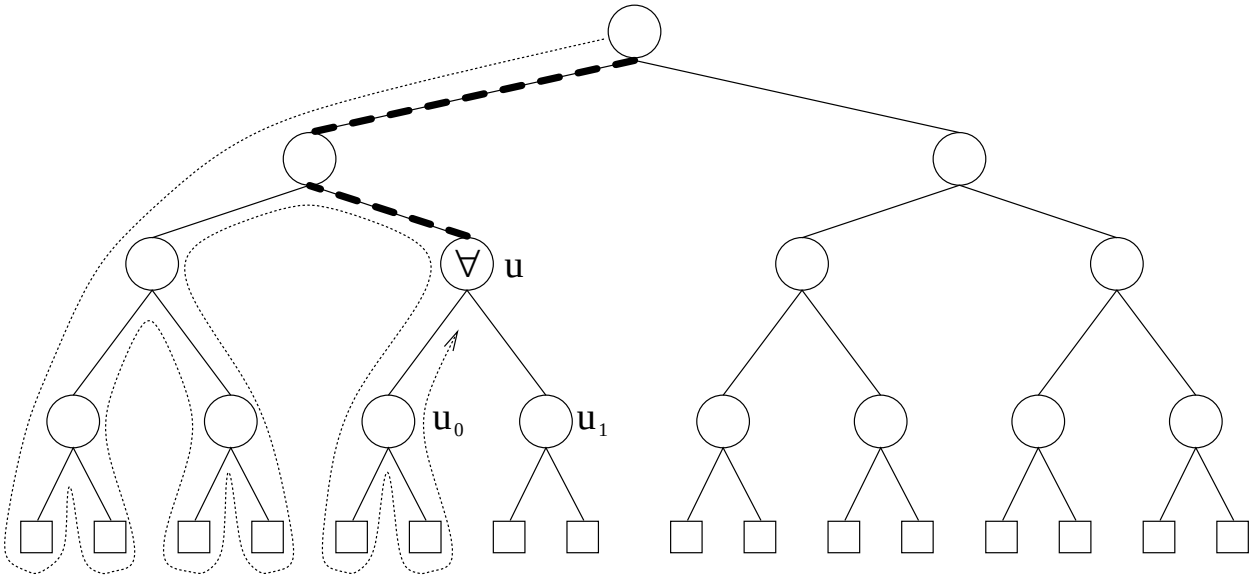

Abbildung 3: Eine Momentaufnahme des Präordnungs-Durchlaufs durch einen Entscheidungsbaum (u mit Söhnen u_0, u_1 der aktuell besuchte Knoten, Ariadne-Faden hervorgehoben).

sich in $2^{cS(n)}$ Rechenschritten in die (oBdA eindeutige) akzeptierende Endkonfiguration K_+ transformieren lässt. Der Schlüssel zum Beweis ist daher die Aussage, dass sich die Relation

$$K \xrightarrow{s} K' :\Leftrightarrow \text{Konfiguration } K \text{ wird von } M \text{ in } 2^s \text{ Rechenschritten in } K' \text{ überführt}$$

durch eine effizient konstruierbare QBF F_s beschreiben lässt. Wir werden im folgenden eine Konfiguration K mit dem entsprechenden binären Codewort einer Länge $m = O(S(n))$ identifizieren. Belegungen einer Kollektion von m Booleschen Variablen, die korrekt gebildeten Codewörtern entsprechen, repräsentieren auf diese Weise Konfigurationen. Wie wir aus dem Beweis des Cook'schen Theorems wissen, existiert eine effizient konstruierbare Boolesche Formel F_0 (in $2m$ Booleschen Variablen) mit $F_0(K, K') = 1$ gdw K und K' Konfigurationen entsprechen (genauer: deren Codewörtern) und wenn $K \xrightarrow{0} K'$ (d.h. K' ist direkte Folgekonfiguration von K). Dies legt (ähnlich wie im Beweis des Theorems von Savich) die folgende Rekursion nahe:

$$F_s[K, K'] := (\exists K'')F_{s-1}[K, K''] \wedge F_{s-1}[K'', K'] \ . \tag{7}$$

Die schlechte Nachricht ist aber, dass die resultierende QBF für $K_0(x) \xrightarrow{cS(n)} K_+(x)$ eine in n exponentielle Länge hat. Dies liegt daran, dass wir in (7) *zwei* rekursive Aufrufe vorfinden. Für $s = cS(n)$ entsteht somit ein vollständig binärer Rekursionsbaum der Tiefe $cS(n)$ mit $2^{cS(n)}$ Blättern.

Einschub Es ist nicht verwunderlich, dass der Plan noch nicht aufgeht. Bisher haben wir ausschließlich $\exists$-Quantoren verwendet. Wenn wir auf diese Weise eine polynomielle

Reduktion erhalten hätten, hätten wir $PSpace = NP$ gezeigt (was eine Sensation wäre). Die noch ausstehende Kompression der Formellänge wird wohl Gebrauch von dem $\forall$-Quantor machen müssen.

Wir ändern jetzt die Rekursionsregel (7) unter Einsatz des $\forall$-Quantors so ab, dass nur noch *ein* rekursiver Aufruf erfolgt. Anstelle eine binären Rekursionsbaumes der Tiefe $cS(n)$ erhalten wir dann einen „Rekursionspfad“ der Tiefe $cS(n)$, der folgerichtig zu einer quantifizierten Booleschen Formel der Kodierungslänge $O(S(n))$ führen wird. In einem gewissen Sinn war bisher alles nur Vorgeplänkel und erst jetzt kommt die

Entscheidende Idee In die Rekursion (7) fügen wir zwei *neue* Variablenkollektionen X, Y ein. Wir werden mit *einem* rekursiven Aufruf testen, ob die Y-Variablenbelegung eine Konfiguration repräsentiert, die durch 2^{s-1} Schritte aus der X-Variablenbelegung hervorgeht. Mit Hilfe eines $\forall$-Quantors werden wir erreichen, dass (X, Y) einmal in der Rolle von (K, K'') und ein weiteres Mal in der Rolle von (K'', K') auftreten. Dadurch ersetzt der *eine* rekursive Aufruf die *beiden* rekursiven Aufrufe in (7).

Die Gleichheit $X = V$ zweier Variablenkollektionen ist komponentenweise zu verstehen. Man beachte auch, dass Gleichheit zweier Boolescher Variablen oder Implikation zwischen Booleschen Variablen als Abkürzungen im Sinne von

$$\begin{aligned} a = b &\Leftrightarrow (a \wedge b) \vee (\bar{a} \wedge \bar{b}) \\ a \to b &\Leftrightarrow \bar{a} \vee b \end{aligned}$$

zu sehen sind. Nach diesen Vorbemerkungen geben wir nun die neue Rekursionsregel an, die (7) ersetzen soll:

$$\begin{aligned} &F_s[K, K'] := \\ &\quad (\exists K'')(\forall X)(\forall Y)\,(((X = V \wedge Y = V'') \vee (X = V'' \wedge Y = V')) \to F_{s-1}[X, Y]) \quad . \end{aligned}$$

Nach etwas kontemplativer Versenkung sollte klar werden, dass die neue Rekursionsregel logisch äquivalent zu (7) ist. Wenn wir jetzt $F_{cS(n)}[K_0(w), K_+]$ gemäß der neuen Rekursion expandieren (und in die Pränexform bringen, so dass die Quantoren ganz am Anfang der Formel stehen), ergibt sich schließlich die angestrebte polynomielle Reduktion von L nach TQBF. **qed.**

Wenn wir uns die Reduktionsabbildung im Beweis von Satz 13.4 nochmals aufmerksam ansehen, können wir die folgende Beobachtung machen:

Bemerkung 13.5 *$x \mapsto F_x$ produziert eine QBF, deren Quantorenkette von x nur indirekt über $|x|$ abhängt. Anders ausgedrückt: verschiedene Wörter x, x' derselben Länge n führen zur selben Quantorenkette in den QBFs F_x und $F_{x'}$.*

Sätze 13.3 und 13.4 liefern zusammen die

Folgerung 13.6 *TQBF ist PSpace-vollständig.*

13.2 Prädikatenlogische Charakterisierung von PSpace

Im folgenden Satz wird *PSpace* mit Hilfe alternierender Quantorenketten charakterisiert. Dabei ist $(\exists)_{pol}$ bzw. $(\forall)_{pol}$ die Abkürzung für einen Quantor der sich über poly(n) Bits erstreckt.

Satz 13.7 *Eine Sprache L gehört genau dann zu PSpace, wenn eine Sprache $L_0 \in P$ und ein Polynom p existieren, so dass*

$$x \in L \Leftrightarrow (\exists y_1)_{pol}(\forall y_2)_{pol}(\exists y_3)_{pol} \cdots (Q_{p(|x|)} y_{p(|x|)})_{pol} \langle y_1, \ldots, y_{p(|x|)}, x\rangle \in L_0 \ . \tag{8}$$

Beweis Wir setzen zunächst $L \in PSpace$ voraus und weisen Bedingung (8) nach. Wegen $L \leq_{pol}$ TQBF gibt es für alle Wörter x eine (aus x polynomiell konstruierbare) Boolesche Formel F_x in $m = \text{poly}(|x|)$ Booleschen Variablen $v_1, \ldots, v_m$ und eine Quantorenkette $\mathcal{Q}_1, \ldots, \mathcal{Q}_m \in \{\exists, \forall\}$ mit der Eigenschaft

$$x \in L \Leftrightarrow (\mathcal{Q}_1 v_1) \ldots (\mathcal{Q}_m v_m) F_x(v_1, \ldots, v_m) \ . \tag{9}$$

Wir erinnern daran, dass die Quantorenkette in (9) von x nur indirekt über $|x|$ abhängt. OBdA gelte $\mathcal{Q}_1 = \exists$. (Ansonsten könnten wir eine redundante Variable mit einem $\exists$-Quantor hinzufügen.) Wir können die Quantorenkette $\mathcal{Q}_1, \ldots, \mathcal{Q}_m$ in maximale Teilketten mit Quantoren nur eines Typs unterteilen. Die Anzahl der Teilketten sei mit $p(|x|)$ bezeichnet. Dann können wir (9) umschreiben wie folgt:

$$x \in L \Leftrightarrow (\exists y_1)_{pol}(\forall y_2)_{pol}(\exists y_3)_{pol} \cdots (Q_{q(|x|)} y_{q(|x|)})_{pol} F_x(y_1, \ldots, y_{p(|x|)}) \ . \tag{10}$$

Hierbei bezeichnet y_i die Teilkollektion der Booleschen Variablen aus $v_1, \ldots, v_m$, welche in der i-ten Teilkette von $(\mathcal{Q}_1 v_1) \ldots (\mathcal{Q}_m v_m)$ vorkommen. Mit

$$L_0 := \{\langle y_1, \ldots, y_{p(|x|)}, x\rangle : F_x(y_1, \ldots, y_{p(|x|)} = 1\} \in P$$

kann schließlich (10) in die Form (8) gebracht werden.
Nehmen wir umgekehrt an, dass L durch Bedingung (8) gegeben ist. Wir haben $L \in PSpace$ zu zeigen. Dieser Nachweis kann in völliger Analogie zum Nachweis von TQBF $\in PSpace$ (Beweis von Satz 13.3) geführt werden. Wir modellieren mit einem Entscheidungsbaum T die möglichen Belegungen der Variablenkollektionen $y_1, \ldots, y_{p(|x|)}$. Wir verwenden wieder eine platzeffiziente EXHAUSTIVE SEARCH durch T unter Einsatz der Ariadne-Faden Technik. Im Laufe der EXHAUSTIVE SEARCH kann eine Markierung der Knoten von T mit Wahrheitswerten vorgenommen werden, so dass x genau dann zu L gehört, wenn die Wurzel dabei mit Wahrheitswert 1 markiert wird. Die Ausfüllung der technischen Details überlassen wir dem Leser und der Leserin. **qed.**

13.3 Weitere PSpace-vollständige Probleme

In diesem Abschnitt nennen wir ohne Beweis ein paar weitere PSpace-vollständige Probleme, um einen Eindruck zu vermitteln, welcher Problemtypus in diese Kategorie gehört.

13.3.1 Probleme mit Automaten und Grammatiken

Definition 13.8 Reguläre Ausdrücke *über einem Alphabet Σ und die von ihnen induzierten formalen Sprachen sind induktiv definiert wie folgt:*

1. *$\emptyset, \epsilon, \sigma$ mit $\sigma \in \Sigma$ sind reguläre Ausdrücke. Die hiervon induzierten Sprachen sind (in dieser Reihenfolge) die leere Menge, die Menge $\{\epsilon\}$ und die Menge $\{\sigma\}$.*
2. *Wenn α, β reguläre Ausdrücke sind, dann sind auch $(\alpha + \beta)$, $(\alpha \cdot \beta)$ und (α^*) reguläre Ausdrücke. Wenn $L(\alpha)$ und $L(\beta)$ die von α und β induzierten Sprachen bezeichnen, dann sind die von $(\alpha+\beta)$, $(\alpha\cdot\beta)$ und (α^*) induzierten Sprachen (in dieser Reihenfolge) $L(\alpha) \cup L(\beta)$, $L(\alpha) \cdot L(\beta)$ und $(L(\alpha))^*$.*

Reguläre Ausdrücke werden zum Beispiel bei Editoren oder in der lexikalischen Analyse von Compilern angewendet. Das folgende Problem fragt danach, ob ein gegebener regulärer Ausdruck eine Sprache induziert, die zumindest ein Wort über dem Grundalphabet ausschließt.

Definition 13.9 *REGULAR EXPRESSION NON-UNIVERSALITY ist das folgende Problem: Zu gegebenem regulären Ausdruck α entscheide ob $L(\alpha) \neq \Sigma^*$.*

Satz 13.10 (Stockmeyer und Meyer, 1973) *Für jedes mindestens zweielementige Grundalphabet (also zum Beispiel für das binäre Alphabet $\{0, 1\}$) gilt: REGULAR EXPRESSION NON-UNIVERSALITY ist PSPACE-vollständig.*

Der Beweis von Stockmeyer und Meyer erfolgt mit einer generischen polynomiellen Reduktion (also ausgehend von einer beliebigen Sprache L aus *PSpace*).

Ein LBA (ausgeschrieben: Linear Bounded Automaton) ist im wesentlichen eine $O(n)$-platzbeschränkte NTM.[5] Es ist bekannt, dass die Klasse der von LBAs akzeptierbaren Sprachen identisch ist mit der Klasse der sogenannten kontextsensitiven Sprachen (die aus der Vorlesung *Theoretische Informatik* bekannt sein könnten).

Definition 13.11 *LBA-ACCEPTANCE ist das Problem zu entscheiden, ob ein gegebener LBA M ein gegebenes Eingabewort x akzeptiert. Hierbei sind also sowohl (eine Kodierung von) M als auch x Bestandteil der Eingabe. LINEAR SPACE ACCEPTANCE ist das entsprechende Problem für $O(n)$-platzbeschränkte DTMs.*

Satz 13.12 (Karp, 1972) *LBA-ACCEPTANCE und LINEAR SPACE ACCEPTANCE sind beides PSpace-vollständige Probleme.*

Der Beweis von Karp, den wir als **Übung** empfehlen, erfolgt über eine (technisch einfache) generische polynomielle Reduktion unter Verwendung eines „padding argument“. Wegen des engen Zusammenhangs zwischen LBAs und kontextsensitiven Grammatiken (auf deren formale Definition wir hier verzichten) ergibt sich leicht die **Übg.**

Folgerung 13.13 *Das Problem zu einer gegebenen kontextsensitiven Grammatik G und einem gegebenen Wort x zu entscheiden, ob x zu der von G induzierten kontextsensitiven Sprache gehört, ist PSpace-vollständig.*

[5] Aus dem Kompressionssatz folgt, dass eine solche NTM oBdA genau n Zellen (also exakt die das Eingabewort enthaltenden Zellen) besucht.

13.3.2 Spielprobleme

Es gibt einen natürlichen Zusammenhang zwischen Zwei-Personen Spielen und alternierenden Quantorenketten. Nennen wir die Personen WEISS und SCHWARZ. Die Frage, ob WEISS (im Anzug) eine Gewinnstrategie über m Züge hat, liest sich in expandierter Form wie folgt:

($\exists$ Zug_1 für WEISS)	($\forall$ Zug_1 von SCHWARZ)
($\exists$ Zug_2 für WEISS)	($\forall$ Zug_2 von SCHWARZ)
...	...
($\exists$ Zug_m für WEISS)	Spielstellung ist gewonnen für WEISS

Die analoge Frage mit WEISS im Nachzug (also SCHWARZ im Anzug) lässt sich analog mit einer alternierenden Quantorenkette formulieren, die mit einem $\forall$-Quantor beginnt. Im Lichte von Satz 13.7 kann es daher nicht überraschen, dass sich über geeignet definierte Spiele PSpace-vollständige Probleme ergeben. Wir konkretisieren dies an folgenden Spielen:

QBF-Spiel: Ein „Spielbrett“ des QBF-Spiels besteht aus einer Booleschen Formel $F = F(v_1, \ldots, v_m)$ in m Booleschen Variablen (wobei verschiedene Spielbretter verschiedene Größen m haben dürfen). Zwei Spieler sind abwechselnd am Zug. Spieler 1 (der „Verifizierer“) hat das Ziel, F zu 1 auszuwerten, wohingegen Spieler 2 (der „Falsifizierer“) die Auswertung 0 anstrebt. Spieler 1 belegt v_1 mit einem Bit seiner Wahl, Spieler 2 danach die Variable v_2, dann wieder Spieler 1 die Variable v_3, und so weiter. Es seien $a_1, \ldots, a_m$ die von den Spielern gewählten Bits. Spieler 1 gewinnt gdw $F(a_1, \ldots, a_m) = 1$.

GENERALIZED HEX: Sei $G = (V, E)$ ein Graph mit zwei ausgezeichneten Knoten $s, t \in V$. Das Spiel HEX[G] ist ein Zwei-Personen Spiel[6], bei welchem WEISS über weiße und SCHWARZ über schwarze Spielsteine verfügt. Beide Spieler ziehen abwechselnd mit WEISS im Anzug. Ein Zug besteht darin, einen eigenen Stein auf einen noch unbesetzten Knoten aus $V \setminus \{s, t\}$ zu setzen. Das Spiel ist vorüber, wenn alle Knoten aus $V \setminus \{s, t\}$ von (weißen oder schwarzen) Steinen besetzt sind. Die finale Spielstellung ist genau dann eine *Gewinnstellung* für WEISS, wenn die Knoten s, t zusammen mit den von weißen Steinen besetzten Knoten einen Verbindungspfad von s nach t in G enthalten. WEISS muss also mit der Strategie spielen, eine Verbindung von s nach t herzustellen; SCHWARZ ist bemüht, solche Verbindungen mit Hilfe von schwarzen Steinen zu vereiteln.

Offensichtlich ist die Frage nach einer Gewinnstrategie für Spieler 1 im QBF-Spiel gerade das Problem TQBF in einer (leicht zu durchschauenden) Verkleidung:

Bemerkung 13.14 *Zu entscheiden, ob Spieler 1 im QBF-Spiel eine Gewinnstrategie hat ist PSpace-vollständig.*

Das folgende Resultat geht in die gleiche Richtung, ist aber weniger offensichtlich:

[6]Für spezielle Graphen in den USA ein handelsübliches Spiel

Satz 13.15 (Even und Tarjan, 1976) *Die Frage, ob WEISS eine Gewinnstrategie im Spiel GENERALIZED HEX besitzt, ist PSpace-vollständig.*

Die Mitgliedschaft in *PSpace* sollte klar sein. Der Beweis der *PSpace*-Härte erfolgt durch Angabe einer polynomiellen Reduktion von TQBF auf GENERALIZED HEX.

In ähnlicher Weise kann man zeigen, dass (mehr oder weniger) natürliche Verallgemeinerungen[7] populärer Spiele (wie zum Beispiel SCHACH oder GO) ebenfalls *PSpace*-hart sind.

Weitere Übungsaufgabe Das japanische Brettspiel „Go-Moku“ wird von zwei Spielern auf einem (19×19)-Gitter gespielt. Die Spieler setzen abwechselnd ihre Steine. Derjenige, der zuerst fünf seiner Steine direkt aufeinanderfolgend in einer Zeile, Spalte oder Diagonale platziert, hat gewonnen. Betrachte dieses Spiel als auf ein $(n \times n)$-Gitter verallgemeinert. Es sei GM die Sprache aller Spielstellungen im verallgemeinerten Go-Moku-Spiel, für die Spieler 1 eine Gewinnstrategie besitzt. **Zeige**, dass GM zur Klasse *PSpace* gehört. **Übg.**

13.4 Offene Probleme

Über die (bisher unbewiesene) $P \neq NP$-Vermutung haben wir schon viel berichtet. Die Vermutung $P \neq PSpace$ ist vergleichsweise schwächer. Im Lichte der diversen prädikatenlogischen Charakterisierungen kann man diese offenen Fragen nun auch folgendermaßen ausdrücken:

P versus NP Sind prädikatenlogische Aussagen[8] die einen $(\exists)_{pol}$-Quantor verwenden dürfen, mächtiger als rein aussagenlogische (also quantorenfreie) Aussagen?

P versus PSpace Sind prädikatenlogische Aussagen, die eine alternierende Quantorenkette variabler Länge verwenden dürfen, mächtiger als rein aussagenlogische Aussagen?

Wir haben die Fragen bewusst informell gehalten, damit die Formulierungen griffig sind. Leider haben die prädikatenlogischen Formulierungen der dahinter liegenden berechnungstheoretischen Probleme bisher nicht wirklich zu einer Klärung der offenen Fragen beigetragen.

[7] Um sinnvoll über Komplexität reden zu können, benötigt man Spielbretter einer variablen Größe.

[8] Gemeint ist stets pränexe Normalform mit Quantoren vom Typ $(\exists)_{pol}$ oder $(\forall)_{pol}$ und einer in Polynomialzeit entscheidbaren Aussage.