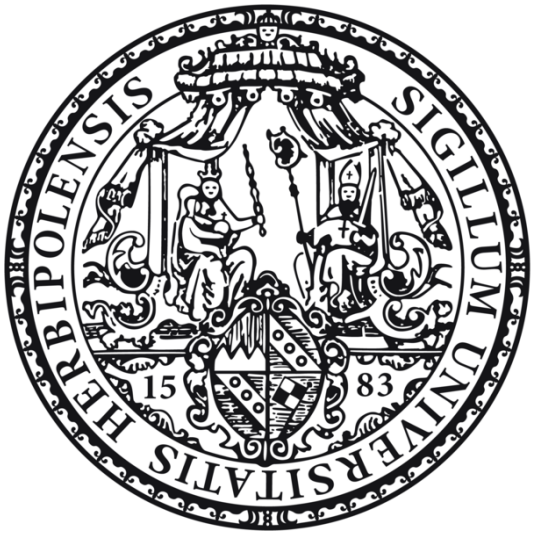
Bachelorarbeit

Über die relativierte Vollständigkeit der Komplexitätsklassen UP versus NP ∩ coNP

Anton Ehrmanntraut

Semester:	Sommersemester 2019
Abgabedatum:	02. Juli 2019
Betreuer:	Prof. Dr. Christian Glaßer



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen, Komplexität und wissensbasierte Systeme

Inhaltsverzeichnis

1 Einleitung

Allgemeiner Gegenstand der Arbeit ist die Untersuchung der Komplexitätsklassen UP und NP ∩ coNP. Intuitiv können die Sprachen der Klasse UP ⊆ NP zunächst, wie im Fall von NP, als Entscheidungsprobleme verstanden werden, die nach der Existenz einer kurzen Lösung zu einem gegebenen Problem fragen. Diese Lösung kann dann in P geprüft werden. Für den Fall UP ergibt sich nun die definierende Einschränkung, dass die Lösung zum Problem auch *eindeutig* sein muss, sollte sie überhaupt existieren. Analog können die Sprachen der bekannteren Klasse NP ∩ coNP wieder als Entscheidungsprobleme verstanden werden, zu der für ein gegebenes Problem nicht nur (a) kurze, einfach überprüfbare Lösungen existieren, sollte das Problem lösbar sein, aber auch (b) analog Widerlegungen existieren, sollte das Problem nicht lösbar sein.

Diese Charakterisierungen macht die Klassen UP und NP ∩ coNP zu Instanzen sogenannter *Promise-Klassen*. Bevor auf diesen Begriff eingegangen wird, folgt die von Valiant aufgestellte präzise Definition von UP. Diese übernimmt die eben angesprochene intuitive Definition und überführt diese auf Turing-Maschinen. (Wie im Fall von NP kann ein akzeptierender Rechenweg als Lösung zum Problem x verstanden werden.)

Definition 1. [HH86; Val76] Sei N eine nichtdeterministische Polynomialzeit-Turing-Maschine. Wir definieren $\mathrm{acc}_N(x)$ als die Anzahl an akzeptierenden Rechenwegen auf der nichtdeterministischen Berechnung $N(x)$.

Die Maschine N ist *kategorisch*, wenn für alle Eingaben x auch $\mathrm{acc}_N(x) \leq 1$. Die Komplexitätsklasse UP entspricht dann jenen Sprachen, die von kategorischen Maschinen erkannt werden können. $\mathrm{UP} =_{\mathrm{df}} \{L(N) \mid N \text{ ist kategorisch}\}$.

Für jede Sprache $L \in \mathrm{UP}$ existiert also *sicher* eine nichtdeterministische Maschine, die nicht nur L entscheidet, aber auch zu jeder Eingabe $x \in L$ auf genau einem Rechenweg akzeptiert. Dieses über die Definition getragene ›Versprechen‹ bzw. *Promise* macht UP zu einer Promise-Klasse. Promise-Klassen können also als Menge an Sprachen charakterisiert werden, wobei jede Sprache jeweils durch eine Maschine M erkannt werden muss. Dabei werden weitere Restriktionen bezüglich Verhalten von M gestellt, die üblicherweise über Zeitschranken oder Determinismus hinaus gehen.

Ähnlich lässt sich auch für NP ∩ coNP ein Promise formulieren, wobei hierfür zwei nichtdeterministische Polynomialzeit-Maschinen notwendig sind. Ein Paar (N_a, N_b) an nichtdeterministischen Polynomialzeit-Turing-Maschinen bezeichnen wir als *komplementär* akzeptierend, wenn $L(N_a) = \overline{L(N_b)}$. Hieraus folgt die analoge Promise-Definition: $\mathrm{NP} \cap \mathrm{coNP} = \{L(N) \mid N_a \text{ und } N_b \text{ akzeptieren komplementär}\}$.

Wie bei allen Komplexitätsklassen stellt sich die Frage nach vollständigen Sprachen. Ob für UP bzw. NP ∩ coNP vollständige Sprachen existieren, ist nicht bekannt. Tatsächlich wurden schon Indizien geliefert, die auf die hohe Schwierigkeit dieser Fragestellung hinweisen. In dieser Arbeit wird deshalb speziell untersucht, ob sich die Vollständigkeit zwischen den Promise-Klassen UP bzw. NP ∩ coNP überträgt. Interesse zur Beantwor-

tung der Frage nach vollständigen Sprachen in UP bzw. $\mathrm{NP} \cap \mathrm{coNP}$ kommt nicht nur aus der Komplexitätstheorie. Im Folgenden werden weitere Aspekte angegeben, welche die Fragestellungen weiter motivieren.

Relevanz in der Kryptographie Die Sicherheit des RSA-Public-Key-Kryptosystems baut auf der Schwierigkeit auf, zu einer zusammengesetzten Zahl nicht in der Lage zu sein, die Primfaktoren effizient zu berechnen. Durch die Multiplikation von Primzahlen wird in RSA dadurch eine sogenannte *Einwegfunktion* implementiert. Einwegfunktionen spielen eine essenzielle Rolle in der Kryptographie, und lassen sich intuitiv als injektive Funktionen verstehen, die ›leicht‹ zu berechnen sind, deren Umkehrung aber nur ›schwer‹ berechenbar. [vgl. GS88, S. 324 für eine formale Def.] Die Umkehrung der RSA-Einwegfunktion ist also die Primfaktorzerlegung, welche bekanntlich eindeutig ist. Eine entsprechendes Entscheidungsproblem liegt damit in UP.

Damit lässt sich die Untersuchung der Klasse UP von kryptographischer Perspektive motivieren. Grollman und Selman zeigen insbesondere, dass genau dann Einwegfunktionen existieren, wenn $\mathrm{P} \neq \mathrm{UP}$. [GS88, Thm. 7] Insbesondere ist $\mathrm{P} = \mathrm{UP}$ ausgeschlossen, wenn UP nicht vollständig ist.

Über die Komplexitätsklasse $\mathrm{NP} \cap \mathrm{coNP}$ lässt sich auch die Sicherheit von allgemeinen Public-Key-Kryptosystemen angeben. Zu einer gegebenen Verschlüsselungsvorschrift E identifizieren Even und Yacobi das *Cracking Problem* als folgende Fragestellung: *Gegeben öffentlicher Schlüssel K, Chiffretext C, finde Klartext M sodass $C = E(K, M)$*. Für das entsprechende Entscheidungsproblem `CP` (*Ist der gesuchte Klartext M kleiner als M'?*) beobachten Even und Yacobi die Zugehörigkeit $\texttt{CP} \in \mathrm{NP} \cap \mathrm{coNP}$. Insbesondere ist ein Kryptosystem nur dann sicher, wenn `CP` nicht effizient entschieden werden kann.

Damit wird die Untersuchung der Klasse $\mathrm{NP} \cap \mathrm{coNP}$ motiviert. Im Fall $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ sind sichere Public-Key-Systeme ausgeschlossen. Gleichzeitig ist nicht erwartet, dass `CP` für ein System NP-hart sein kann. [vgl. EY80] Insbesondere existieren dann sichere Public-Key-Systeme, wenn keine vollständigen Sprachen für $\mathrm{NP} \cap \mathrm{coNP}$ existieren. Nichtsdestotrotz kann von ›maximal sicheren‹ Systemen E gesprochen werden, wenn $\mathrm{NP} \cap \mathrm{coNP}$ vollständige Mengen zulässt. (Das heißt das entsprechende `CP` ist vollständig für den Schnitt.)

Motivation durch aussagenlogische Beweissysteme Cook und Reckhow starteten die systematische Untersuchung von sogenannten *aussagenlogischen Beweissystemen*. Ohne näher auf die Details einzugehen, können solche Beweissysteme als Funktionen verstanden werden, die in Polynomialzeit prüfen, ob gegebener Beweis w die aussagenlogische Formel φ beweist. Sinnvollerweise existiert dann auch zu jeder Tautologie φ ein Beweis, damit das Beweissystem (semantisch) vollständig ist. [vgl. DG19; vgl. Pud17]

Zwar lassen sich solche Beweissysteme angeben, zurzeit ist jedoch kein System bekannt, in der zu allen Tautologien φ der Beweis höchstens polynomiell länger als φ

ist. Cook und Reckhow beobachteten, dass ein solches polynomiell beschränktes Beweissystem genau dann existiert, wenn die Komplexitätsklasse NP unter Komplement abgeschlossen ist, bzw. NP = coNP.

Diese Einsicht motivierte das sogenannte *Cook–Reckow-Programm*: Man nähert sich der Frage NP $\neq$ coNP durch Untersuchen immer stärker Beweissysteme. In diesem Sinn wurden Konzepte wie (p-)Simulation und (p-)Optimalität eingeführt, die als Analog zu den komplexitätstheoretischen Konzepten der (Polynomialzeit-)Reduktion und (Polynomialzeit-)Vollständigkeit verstanden werden können. [KMT03]

In seiner Arbeit »Incompleteness in the finite domain« [Pud17] geht Pudlák der Frage nach, in welchen Beziehungen die Mächtigkeit logischer Axiomensysteme und die Komplexitätstheorie stehen. Er gibt Überblick über die Implikationen zwischen Hypothesen betreffend der Existenz optimaler Beweissysteme, Separationen von Komplexitätsklassen, und insbesondere der Existenz vollständiger Sprachen für ausgewählte Promise-Klassen. Darunter befinden sich auch UP und NP $\cap$ coNP, womit die Untersuchung der Frage nach vollständigen Sprachen für diese Klassen noch weiter motiviert wird. Pudlák definiert unter anderem folgende Hypothesen:

Definition 2. (i) Die Hypothese UP ist definiert als die Aussage, dass *keine vollständigen Mengen bezüglich Many-One-Reduktion in* UP *existieren.*

(ii) Die Hypothese NP$\cap$coNP ist definiert als die Aussage, dass *keine vollständigen Mengen bezüglich Many-One-Reduktion in* NP $\cap$ coNP *existieren.*

(iii) Die Hypothese CON (bzw. SAT) ist definiert als die Aussage, dass *keine p-optimalen Beweissysteme für die Menge* TAUT *der aussagenlogischen Tautologien (bzw. für die Menge* SAT *der erfüllbaren aussagenlogischen Formeln) existieren.*

Abbildung 1 fasst ausgewählte Implikationen zwischen den Hypothesen zusammen. Insbesondere ist noch weiter die von UP implizierte Hypothese CON hervorzuheben: Neben den bereits eingezeichneten Konsequenzen der Hypothese bezüglich Separationen an Komplexitätsklassen, steht die Hypothese in enger Verbindung zur *Endlichen Widerspruchsfreiheit* von Axiomensystemen.

Die Endliche Widerspruchsfreiheit eines Axiomensystems T kann hier als Aussage verstanden werden, dass durch Beweise der Länge $\leq n$ aus den Axiomen von T kein Widerspruch herleitbar ist. Krajíček und Pudlák zeigen, dass $\neg$CON äquivalent zur Existenz eines Axiomensystem S ist, welches in polynomieller Zeit (abh. von n) ein Beweis in S konstruiert werden kann, der die endliche Widerspruchsfreiheit (begrenzt auf n) eines beliebigen Axiomensystems T beweist. Ein solcher Beweis der endlichen Widerspruchsfreiheit kann verwendet werden, um eine ›finite‹ Variante des *Hilbertschen Programms* zu realisieren. Insbesondere wäre dann die Formulierung eines starken Axiomensystems T möglich, welches *bewiesen* widerspruchsfrei ist, wenn man sich auf angemessen kurze Beweise beschränkt. [vgl. KP89; Dos19]

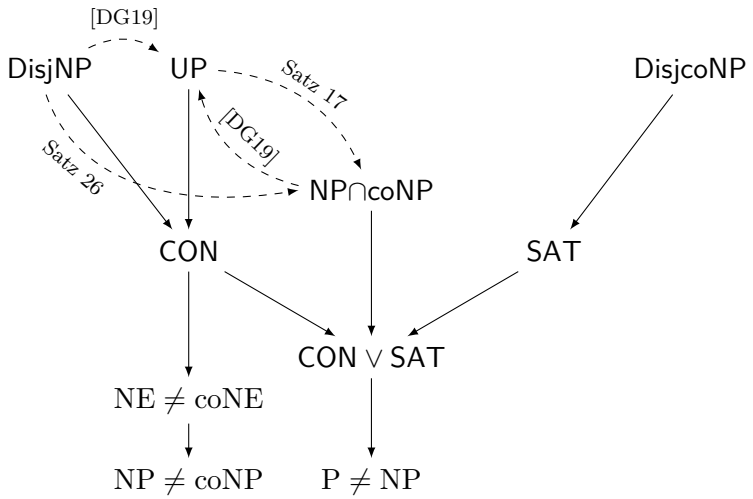


Abbildung 1: Unvollständige Abbildung der Beziehungen zwischen Pudláks Hypothesen aus Definition 2. Die Hypothesen Disj(co)NP entsprechen der Nichtexistenz von vollständigen Paaren für Disj(co)NP und werden in Kapitel 5 erläutert. Durchgezogene Pfeile von A zu B geben an, dass (relativierbar) $\mathsf{A} \Rightarrow \mathsf{B}$ gilt. [vgl. Pud17, S. 436] Unterbrochene Pfeile geben an, dass relativ zu einem Orakel $\mathsf{A} \wedge \neg \mathsf{B}$ gilt. (Erläutert im folgenden Kapitel.)

Aufbau der Arbeit Die Beantwortung der offenen Frage, ob nun tatsächlich (keine) vollständigen Sprachen für UP bzw. NP ∩ coNP existieren, wird wahrscheinlich nicht mit aktuell bekannten Beweismethoden zu zeigen sein, sondern nur mit anspruchsvollsten *nichtrelativierbaren* Beweismethoden. Diese Vermutung stützt sich dabei auf sogenannte *Orakel*, die für diese Fragestellungen speziell konstruiert wurden.

Orakel geben uns die Möglichkeit, schwierige Fragestellungen auf ›alternative Universen‹ zu übertragen, zu *relativieren*. Dort hat jede Turing-Maschine Zugriff auf das Orakel, welches Fragen der Turing-Maschine *unverzüglich ohne Rechenaufwand korrekt* beantwortet. Folgendes Kapitel 2 definiert den Begriff des Orakels, der Orakel-Turing-Maschine, und der (nicht-)relativierbaren Aussagen präzise. Anschließend wird erklärt, inwiefern spezielle Orakel als Indiz verstanden werden können, dass Fragestellungen der Komplexitätstheorie nicht über die ›einfacheren‹ relativierbaren Methoden zu beantworten sind.

Die nächsten beiden Kapitel wenden sich der primären Aufgabenstellung der Arbeit zu, und gehen der Fragestellung nach, inwiefern sich Vollständigkeit zwischen den Promise-Klassen UP und NP ∩ coNP übertragen lässt. Über zwei Konstruktionen von geeigneten Orakeln werden auf ähnliche Weise Indizien geliefert, welche auf die Schwierigkeit des Beweises der folgenden möglichen Implikationen $\mathsf{UP} \Rightarrow \neg\mathsf{NP} \cap \mathsf{coNP}$ bzw. $\mathsf{UP} \Rightarrow \mathsf{NP} \cap \mathsf{coNP}$

hinweist. Kapitel 3 wendet sich der ersten der genannten Implikationen zu, und kombiniert hierbei existierende Resultate von Sipser, Hartmanis und Hemachandra.

Die Konstruktion in Kapitel 4 für die zweite Implikation ist erheblich aufwändiger, und stützt sich auf algorithmische Ideen von Baker, Gill und Solovay. Hierfür wurde das von ihnen vorgestellte Verfahren für den hier untersuchten Fall abgeändert.

Aufbauend auf dieser Modifikation des Verfahrens wird in Kapitel 5 versucht, das Verfahren auf verwandte Fragestellungen zu verallgemeinern. Die Hypothesen $\mathsf{Disj(co)NP}$ werden erläutert, und eine Ergänzung des Verfahrens angegeben, welches ›leichte‹ relativierbare Beweise der Implikation $\mathsf{UP} \wedge \mathsf{DisjNP} \Rightarrow \neg\mathsf{NP}\cap\mathsf{coNP}$ ausschließt. Am Beispiel der Hypothese $\mathsf{DisjcoNP}$ werden auch die Grenzen des Verfahrens deutlich gemacht.

In Kapitel 6 werden die Ergebnisse der Arbeit zusammengetragen. Insbesondere ergibt sich mit Ergebnissen von Dose und Glaßer die paarweise Unabhängigkeit der beiden Hypothesen UP und $\mathsf{NP}\cap\mathsf{coNP}$ unter relativierbaren Beweismethoden. Abschließend werden mögliche Interpretationen der Grenzen des Verfahrens im Hinblick auf die Hypothese $\mathsf{DisjcoNP}$ diskutiert.

2 Grundlagen

Wir fixieren Alphabet $\Sigma =_{\text{df}} \{0,1\}$ für den Rest der Arbeit. Das Wort $\varepsilon \in \Sigma^*$ bezeichnet das leere Wort. Die Länge eines Wortes $w \in \Sigma^*$ wird mit $|w|$ bezeichnet, Wörter mit Suffix 0 werden als gerade, mit Suffix 1 als ungerade bezeichnet. Für endliche Mengen A bezeichnet $|A|$ die Anzahl Elemente der Menge. Sei für $A \subseteq \Sigma^*$ die Menge $A^{=n}$ als alle Wörter aus A mit Länge n definiert, also $A^{=n} =_{\text{df}} \{w \mid w \in A, |w| = n\}$. Das Komplement $\overline{A}$ ist hier als $\Sigma^* \setminus A$ zu verstehen.

Die Menge $\mathbb{N}$ ist hier als die Menge natürlicher Zahlen mit 0 zu verstehen. Die Menge aller Wörter Σ^* sind bekanntlich bijektiv zu $\mathbb{N}$, deshalb können wir jede Zahl als Wort interpretieren und andersherum. Insbesondere meint für $n \in \mathbb{N}$ der Ausdruck $|n|$ die Länge des von n repräsentierten Wortes. Die Ausdrücke 0^n und 1^n sind als Wörter aus Σ^n zu verstehen.

Während der Arbeit werden wir regelmäßig von Listencodierungen Gebrauch machen. Diese ordnet eine endliche Folge an Wörtern $w_1, w_2, \ldots, w_n$ aus Σ^* ein Listenwort $\langle w_1, w_2, \ldots, w_n \rangle$ aus Σ^* zu. Insbesondere existiert dann zu einer Liste c genau ein $n \in \mathbb{N}$, und eindeutige $w_1, \ldots, w_n$ sodass $c = \langle w_1, \ldots, w_n \rangle$. Endliche Mengen von endlich großen Wörtern lassen sich dadurch auch als ein Wort codieren. Eine solche Listencodierung lässt sich beispielsweise über $\langle w_1, \ldots, w_n \rangle \mapsto f(\#w_1\# \cdots \#w_n\#)$ angeben. Hierbei ist f eine Codierung vom Alphabet $\{0, 1, \#\}$ auf Σ mit $\{0 \mapsto 00, 1 \mapsto 11, \# \mapsto 01\}$. Die Codierung $\langle \cdots \rangle$ ist damit polynomialzeit-berechnbar und -invertierbar.

Die Folge $(p_{(i)})_{i \in \mathbb{N}}$ ist eine injektive Aufzählung aller abzählbar unendlich vielen Primzahlen. Zur Vereinfachung der folgenden Beweise definieren wir uns überschneidungsfreie Aufzählungen $(p_j)_{j \in \mathbb{N}}$ und $(p_{a,b})_{a,b \in \mathbb{N}}$. Sei hierfür $p_j =_{\text{df}} p_{(g(j))}$, $p_{a,b} =_{\text{df}} p_{(g(a,b))}$, wobei $g \colon \mathbb{N} \cup \mathbb{N}^2 \to \mathbb{N}$ eine Bijektion ist. Insbesondere ist dann (a) $p_j \neq p_{a,b}$ für alle $j, a, b \in \mathbb{N}$, (b) $p_j \neq p_{j'}$ für alle $j \neq j'$, und (c) $p_{a,b} \neq p_{c,d}$ für alle $(a, b) \neq (c, d)$.

Die Folge $(t_i)_{i \in \mathbb{N}}$ ist eine Aufzählung an Polynomen mit $t_i(n) =_{\text{df}} n^i + i$. Insbesondere existiert dann für jedes Polynom p ein $i \in \mathbb{N}$ sodass $p(n) \leq t_i(n)$ für alle $n \in \mathbb{N}$. Die Funktion $\log \colon \mathbb{N} \to \mathbb{N}$ ist definiert als $\lfloor \log_2(n) \rfloor$ für $n > 0$, ansonsten 0.

Im Folgenden sind Turing-Maschinen immer als *nichtdeterministische Orakel*-Turing-Maschinen zu verstehen. Das Orakel O ist hierbei eine Teilmenge von Σ^*. Allen Maschinen M steht bei der Berechnung $M(x)$ ein Orakelband zur Verfügung, auf denen von der Maschine Fragen $z \in \Sigma^*$ geschrieben werden können. Erreicht M einen ausgezeichneten Fragezustand $q_?$, nimmt die Maschine auf der Berechnung $M^O(x)$ mit Orakel O im nächsten Schritt den Zustand q_{yes} an, wenn $z \in O$, ansonsten q_{no}. Damit ist die Berechnung der Maschinen von Eingabe *und* dem Orakel abhängig. Zur Berechnung $M^O(x)$ sagen wir auch, dass $M(x)$ *relativ zu* O berechnet wird. Berechnungen $M(x)$ ohne angegebenes Orakel sind als Berechnungen mit leerem Orakel $O = \emptyset$ zu verstehen.

Allen Turing-Maschinen steht ein Ausgabeband zur Verfügung, und jede Turing-Maschine spezifiziert eine Teilmenge *akzeptierender Zustände*. Eine Maschine berechnet für jedes Orakel $O \subseteq \Sigma^*$ daher sowohl eine *Sprache* als auch eine *Funktion*:

- Die von M^O entschiedene Sprache ist wie üblich definiert als $L(M^O) =_{\text{df}} \{x \mid x \in \Sigma^*$, ein Rechenweg von $M^O(x)$ hält mit akzeptierenden Zustand$\}$ definiert. Der Inhalt des Ausgabebands ist hierbei irrelevant.
- Arbeitet die Maschine M^O deterministisch, berechnet sie für alle $n > 0$ eine Funktion $M^O\colon (\Sigma^*)^n \to \Sigma^*$. Hierbei ist $M^O(x_1, \ldots, x_n)$ der Inhalt des Ausgabebands auf Eingabe $\langle x_1, \ldots, x_n\rangle$ in der haltenden Konfigurationen. Insbesondere ist der Funktionswert nur definiert, wenn M relativ zu O auf diese Eingabe hält. Ob die Berechnung in einem akzeptierenden Zustand hält, ist hierbei irrelevant.

Sprechen wir von *Transduktoren*, dann sind deterministische Polynomialzeit-Maschinen T gemeint, deren berechnete Funktion $T^O\colon \Sigma^* \to \Sigma^*$ für alle $O \subseteq \Sigma^*$ total ist. Da das Akzeptanzverhalten einer Maschine vom Orakel abhängig ist, bezeichnen wir Maschinen nur kategorisch (bzw. ein Maschinepaar komplementär) im Zusammenhang mit einer Angabe eines Orakels.

Die Komplexitätsklasse P^O (bzw. FP^O, NP^O) ist dann wie P definiert, nur dass alle Maschinen relativ zu O rechnen. Die Zeit zur Beantwortung der Orakelfragen wird dabei insbesondere nicht mitgezählt. Analog ist $\leq_{\mathrm{m}}^{\mathrm{p},O}$-Reduktion wie $\leq_{\mathrm{m}}^{\mathrm{p}}$-Reduktion definiert, nur dass die Reduktionsfunktion von einem Transduktor mit Zugriff auf Orakel O berechnet werden muss.

2.1 Aufzählungen von Turing-Maschinen

Jede Maschine M kann injektiv durch ein Codewort $\langle M\rangle$ aus Σ^* repräsentiert werden, indem beispielsweise Zustandsmenge, Frage-, Antwort-, und akzeptierende Zustände, sowie Übergangsrelation von M entsprechend in $\langle M\rangle$ codiert werden. Die Codierung von M ist hierbei so effizient, dass in polynomieller Zeit Zustandsmenge, Übergangsrelation, etc. bestimmt werden können.

Mittels einer solchen Codierung lässt sich nun eine *Standard-Aufzählung* $(P_i)_{i\in\mathbb{N}}$ aller deterministischer Polynomialzeit-Maschinen angeben. Diese Aufzählung sichert drei Eigenschaften für alle $O \subseteq \Sigma^*$:

- Für alle $i \in \mathbb{N}$ entscheidet P_i eine Sprache aus P, das heißt $L(P_i^O) \in \mathrm{P}^O$. Insbesondere ist P_i deterministisch und die Laufzeit von $P_i^O(x)$ auf $t_i(|x|) = |x|^i + i$ beschränkt.
- Für jede Sprache L aus P^O existiert $i \in \mathbb{N}$ sodass $L = L(P_i^O)$.
- Die Maschinen P_i sind effizient in Polynomialzeit simulierbar.

Wir konstruieren im folgenden Lemma eine solche Aufzählung, sowie auch (T_i) für Transduktoren und (N_i) für nichtdeterministische Maschinen. Diese Standard-Aufzählungen verwenden wir im gesamten Rest der Arbeit.

Lemma 3. [vgl. DG19, S. 5] *Es existieren* Standard-Aufzählungen $(P_i)_{i\in\mathbb{N}}$, $(T_i)_{i\in\mathbb{N}}$, $(N_i)_{i\in\mathbb{N}}$ *sodass für alle* $O \subseteq \Sigma^*$

(i) Die Laufzeit von $P_i^O(x)$, $T_i^O(x)$ *und* $N_i^O(x)$ *ist auf* $t_i(|x|)$ *beschränkt. Damit sind insbesondere die Anzahl und Länge der Orakelfragen einer Rechnung von* $P_i(x)$, $T_i^O(x)$ *und* $N_i^O(x)$ *auf* $t_i(|x|)$ *beschränkt.*

(ii) $\{L(P_i^O) \mid i \in \mathbb{N}\} = \mathrm{P}^O$, $\{T_i^O \mid i \in \mathbb{N}\} = \mathrm{FP}^O$, $\{L(N_i^O) \mid i \in \mathbb{N}\} = \mathrm{NP}^O$.

(iii) Es existiert eine deterministische Polynomialzeit-Maschine U_P^O *die auf Eingabe* $\langle i, 1^{t_i(|x|)}, x\rangle$ *das Akzeptanzverhalten der Berechnung von* $P_i^O(x)$ *simuliert, also genau dann akzeptiert wenn* $P_i^O(x)$ *akzeptiert. Analog existiert eine nichtdeterministische Polynomialzeit-Maschine* U_N^O. *Analog existiert auch ein Transduktor* U_T^O, *für den* $U_T^O(i, 1^{t_i(|x|)}, x) = T_i^O(x)$ *gilt.*

Beweis. Wir geben eine solche Standard-Aufzählung für Maschinen P_i an, die anderen Aufzählungen lassen sich analog angeben. Sei für $i \geq 1$ im Folgenden $M_{\text{clock},i}$ eine deterministische k'-Band-Maschine, die auf Eingaben der Länge n nach genau $t_i(n) - n$ hält.

Wir definieren für $i \in \mathbb{N}$ die Maschine P_i folgendermaßen: Ist $i < 2$, oder codiert i keine deterministische Maschine, entspricht P_i der sofort haltenden Maschine. Ansonsten, wenn M eine deterministische k-Band-Maschine ist, definieren wir Maschine P_i mit $k + k' + 1$ Bändern über folgende Arbeitsweise auf Eingabe x:

(1) Kopiere die Eingabe auf das Band $k + 1$ (in $|x|$ Schritten).

(2) Führe die Berechnungen von M und $M_{\text{clock},i}$ unabhängig und parallel aus. Hierbei verwendet M die Arbeitsbänder 1 bis k, $M_{\text{clock},i}$ Band $k+1$ als Eingabe-, sowie $k+2$ bis $k + k' + 1$ als Arbeitsbänder. (Das kann durch Ergänzen der Übergangsrelation erreicht werden.)

(3) Hält die Berechnung von $M_{\text{clock},i}$ vor der Berechnung von M, lehne x ab.

(4) Ansonsten, akzeptiere x entsprechend dem Haltezustand von M.

Zu (i): Die parallel durchgeführte Berechnung von $M_{\text{clock},i}$ begrenzt die Berechnugn von $P_i^O(x)$ auf $t_i(|x|)$ Takte, damit ist die Eigenschaft (i) schon gesichert. Insbesondere können in $t_i(|x|)$ Takten auch nur höchstens so viele Orakelfragen gestellt werden. Das Schreiben einer Frage der Länge $t_i(|x|)$ benötigt mindestens entsprechend so viel Zeit, also ist das eine obere Schranke der Fragelänge auf der Berechnung von $P_i(x)$.

Zu (ii): Nach Definition handelt es sich bei allen P_i um deterministische Maschinen, deren Laufzeit durch ein Polynom beschränkt ist. Damit ist $\{L(P_i^O) \mid i \in \mathbb{N}\} \subseteq \mathrm{P}^O$.

Wir zeigen nun, dass für jedes $L \in \mathrm{P}^O$ ein $i' \in \mathbb{N}$ existiert, sodass $L = L(P_{i'}^O)$. Nach Wahl von L existiert eine deterministische Turing-Maschine M mit polynomieller Laufzeitschranke p. Wir haben schon beobachtet, dass $p(|x|) + |x| \leq t_i(|x|)$ für hinreichend großes i.

Nun modifizieren wir M zu M', indem wir so lange neue Zustände hinzufügen, bis $i' = \langle M' \rangle \geq i$. Insbesondere verhält sich M' wie M. Damit ist nach Definition auch M' auf Laufzeit $p(|x|) + |x| \leq t_i(|x|) \leq t_{i'}(|x|)$ beschränkt. Die definierte Maschine $P_{i'}$ kann also M' vollständig in $t_{i'}(|x|) - |x| \geq p(|x|)$ Schritten simulieren; es gilt $L = L(M^O) = L(M'^O) = L(P_{i'}^O)$.

Also $\{L(P_i^O) \mid i \in \mathbb{N}\} \supseteq \mathrm{P}^O$, damit sind die Mengen gleich, (ii) gezeigt.

Zu (iii): Wir machen Gebrauch von einer universellen deterministischen Turing-Maschine U, die auf Eingabe $\langle \langle M \rangle, x \rangle$ die deterministische Berechnung $M(x)$ simuliert. Diese existieren mit polynomiellen Overhead, das heißt m Takte der der Berechnung von $M(x)$ werden in Zeit $\mathcal{O}(|i|^2 \cdot m^2)$ simuliert.

Weiterhin können wir davon ausgehen, dass die Funktion $i \mapsto \langle P_i \rangle$, in deterministischer Polynomialzeit (im Folgenden $\mathcal{O}(|i|^p)$) berechenbar ist. Insbesondere ist der Test, ob i eine deterministische Maschine codiert, durch Scan der codierten Übergangsrelation in Linearzeit möglich.

Dann lässt sich folgende Arbeitsweise für die Maschine U_P^O auf Eingabe $\langle i, u, x \rangle$ mit Länge n angeben:

> Berechne zunächst $\langle P_i \rangle$ aus i. Führe anschließend die Berechnung von $U(\langle P_i \rangle, x)$ aus. Orakelfragen während der Berechnung werden von U an das Orakel O weitergeleitet. Akzeptiere x entsprechend.

Da die Laufzeit von P_i nach Konstruktion auf t_i beschränkt ist, ist die Laufzeit von U_P^O auf $\mathcal{O}(|i|^p + (|i|^{2p} \cdot t_i(|x|)^2))$ beschränkt. Da $|i|, t_i(|x|) \leq n$ also auf ein Polynom beschränkt. □

Standard-Aufzählungen und vollständige Sprachen Neben der Definition unserer Standard-Aufzählungen (P_i), (N_i) haben wir in Lemma 3(iii) auch gesehen, dass beliebige P_i, N_i *effizient* simuliert werden. Das ermöglicht für P und NP insbesondere die Angabe von künstlichen vollständigen Mengen.

Beispielsweise lässt sich folgende $\leq_{\mathrm{m}}^{\mathrm{p}}$-vollständige Menge für NP angeben.

$$K =_{\mathrm{df}} \{\langle i, 1^{t_i(|x|)}, x \rangle \mid \text{Simulation } U_N(i, 1^{t_i(|x|)}, x) \text{ akzeptiert}\}.$$

Die Sprache ist in nichtdeterministischer Polynomialzeit entscheidbar, ist ja U_N eine nichtdeterministische Polynomialzeit-Maschine (mit Laufzeit in Abhängigkeit von $n = |\langle i, 1^{t_i(|x|)}, x \rangle|$). Gleichzeitig ist K vollständig. Sei $A \in \mathrm{NP}$ beliebig, und $A = L(N_i)$, nach Lemma 3(ii). Dann ist mit $f \colon x \mapsto \langle i, 1^{t_i(|x|)}, x \rangle$ offenbar eine FP-Reduktion gegeben, und

$$x \in A \;\Leftrightarrow\; N_i(x) \text{ akz.} \;\Leftrightarrow\; U_N(i, 1^{t_i(|x|)}, x) \text{ akz.} \;\Leftrightarrow\; \langle i, 1^{t_i(|x|)}, x \rangle \in K \;\Leftrightarrow\; f(x) \in K,$$

also die Reduktion korrekt. [vgl. Per14, prop. 3-M]

Eine solche Angabe einer vollständigen Menge ist insbesondere nicht bei den diskutierten Promise-Klassen UP und NP ∩ coNP möglich. Prinzipiell lässt sich beispielsweise für UP eine Aufzählung an kategorischen Maschinen (UP_i) *definieren*, sodass $\{L(UP_i) \mid i \in \mathbb{N}\} = \text{UP}$, und Eigenschaften (i) und (ii) des Lemma 3 erfüllt sind. Insbesondere aber ist Eigenschaft 3(iii), also die Simulation von beliebigen UP_i nicht in UP berechenbar; eine universelle Maschine U_{UP}, wie gefordert wird, ist nicht vorhanden. (Bzw. existiert kein solcher Beweis für dessen Existenz.) Damit wäre eine analoge Sprache K nicht in UP entscheidbar, also nicht vollständig für UP.

Die Schwierigkeit einer Angabe U_{UP} ergibt sich aus dem Promise von UP: Wollte man ein Algorithmus für den Simulator U_{UP} angeben, müsste geprüft werden, ob die von i codierte Maschine M den Promise erfüllt, also ob M kategorisch arbeitet. Im Gegensatz zum Simulator U_P ist diese Entscheidung nicht trivial; bei U_P reichte es aus die Übergangsrelation von M zu prüfen.

Zu einem gegebenen beliebigen Maschinencode $\langle M \rangle$ ist bislang nicht einmal eine NP-berechenbare Möglichkeit bekannt, ob M kategorisch arbeitet. Aus dieser Schwierigkeit eröffnet sich die Hypothese UP, also ob die Klasse UP tatsächlich überhaupt (keine) vollständigen Sprachen besitzt. [vgl. Wec00, S. 179]

2.2 Diagonalisierung, Relativierbare Aussagen

Diagonalisierung zum Separieren von Komplexitätsklassen Schon die eben definierte Standard-Aufzählung an Maschinen ermöglicht, zusammen mit effizienter Simulation, das Beweisen von echten Inklusionen zwischen gewissen Klassen. Möchte man beispielsweise $\mathcal{C} \subsetneq \mathcal{D}$ zeigen, müsste man eine *Zeugensprache* L in der Klasse $\mathcal{D}$ angeben, die von keiner Maschine in $\mathcal{C}$ entschieden werden kann; für jede Maschine in $\mathcal{C}$ wird eine Eingabe x akzeptiert genau dann wenn $x \notin L$.

Lassen sich alle Maschinen $M_0, M_1, \ldots$ der Klasse $\mathcal{C}$ aufzählen und simulieren, so lässt sich die Beweismethode der *Diagonalisierung* anwenden, um eine solche Menge L zu konstruieren. Für *jede* mögliche Maschine M_i, die eine Sprache in $\mathcal{C}$ entscheidet, wählt man ein x_i und setzt $x_i \in L$ genau dann wenn die Simulation von $M(x_i)$ ablehnt. Ist nun die Simulation von jedem $M_i(x_i)$ in den Schranken der Klasse $\mathcal{D}$ möglich, ist offenbar auch $L \in \mathcal{D}$: Auf Eingabe x_i wird $M_i(x_i)$ simuliert, und akzeptiert genau dann wenn die Simulation ablehnt. [vgl. Per14, S. 169]

Obwohl die im vorigen Abschnitt angegebene universelle Maschine U_P relativ ineffizient ist, können wir damit schon exemplarisch den ›Hierarchiesatz‹ $\text{P} \subsetneq \text{EXP}$ zeigen.

Beobachtung 4. $\text{P} \subsetneq \text{EXP}$.

Beweis. Wir definieren $L =_{\text{df}} \{1^i \mid U_P(i, 1^{t_i(i)}, 1^i) \text{ lehnt ab}\}$, und zeigen $L \in \text{EXP} \setminus \text{P}$. Die Entscheidung $1^i \in L$ ist hierbei auf polynomielle Zeit abhängig von i bzw. Eingabelänge beschränkt, nach Lemma 3(iii). Insbesondere existiert dann geeignetes c, sodass für jedes i die Entscheidung in $2^{(i^c+c)}$ also in Exponentialzeit möglich ist. Damit $L \in \text{EXP}$.

Gleichzeitig $L \notin \mathrm{P}$. Wir nehmen das Gegenteil an, dann existiert nach Lemma 3(ii) ein P_i mit $L = L(P_i)$. Dann gilt

$$P_i(1^i) \text{ akz.} \Leftrightarrow 1^i \in L \Leftrightarrow U_P(i, 1^{t_i(i)}, 1^i) \text{ lehnt ab} \Leftrightarrow P_i(1^i) \text{ lenht ab.}$$

Widerspruch, Annhame war falsch, also $L \in \mathrm{EXP} \setminus \mathrm{P}$. □

Relativierbare Aussagen und Grenzen der Diagonalisierung Solche einfachen Beweise über Diagonalisierungen führen hierbei Simulationen von Maschinen gewissermaßen als Blackbox aus. Die (möglicherweise sehr komplexe) Funktionsweise der diagonalisierten Maschinen wird in solchen Beweisen nicht beachtet. So lässt sich der eben gezeigte Hierarchiesatz auf ein beliebiges Orakel O *relativieren*: Mit nur minimalen Modifikationen am Beweis können wir zeigen, dass für alle $O \subseteq \Sigma^*$ auch $\mathrm{P}^O \subsetneq \mathrm{EXP}^O$ gilt. Im angegebenen EXP-Algorithmus des Beweises müssen nur die im Rahmen der Simulation von $P_i^O(1^i)$ gestellten Orakelfragen ›weitergeleitet‹ werden. Der EXP-Algorithmus hat ja nun die Möglichkeit, Orakelfragen $z \in O$ zu stellen.

Fast alle bekannten Beweistechniken der Komplexitätstheorie zeigen Aussagen, sodass auch diese *relativierbar* sind. Also auch dann gelten, wenn alle Turing-Maschinen relativ zu einem beliebigem Orakel O rechnen. Aus dieser Beobachtung ergibt sich die hohe Relevanz von Orakelkonstruktionen: Sei hierfür O ein konstruiertes Orakel, relativ zu dieser eine Aussage A gilt. (Die in A diskutierten Maschinen haben also Zugriff zu O.) Durch dieses Orakel ist nun ein großes Indiz für die Schwierigkeit eines Beweises für $\neg\mathsf{A}$ geliefert. Insbesondere kann $\neg\mathsf{A}$ dann nicht durch relativierbare bekannte Beweistechniken wie einfache Diagonalisierungen bzw. Maschinensimulationen gezeigt werden. Es ergebe sich sonst wegen Relativierbarkeit folgender Widerspruch:

$$\begin{aligned} \forall O' : (\neg\mathsf{A} \text{ gilt relativ zu } O') \Rightarrow \neg\mathsf{A} \text{ gilt relativ zu } O, \\ \text{aber auch} \quad \mathsf{A} \text{ gilt relativ zu } O. \end{aligned}$$

So gaben beispielsweise Baker, Gill und Solovay Orakel A und B an, relativ zu diesen $\mathrm{P}^A = \mathrm{NP}^A$, $\mathrm{P}^B \neq \mathrm{NP}^B$ und gaben damit Indizien, dass die Antwort auf die P-NP-Frage nur mit nichtrelativierbaren Methoden zu zeigen ist. [BGS75] Analoge Resultate existieren für UP und $\mathsf{NP} \cap \mathsf{coNP}$. Auch diese Hypothesen sind nur mit nichtrelativierbaren Methoden beweisbar bzw. widerlegbar, was die in der Einleitung angesprochene Schwierigkeit der Beantwortung dieser Hypothesen nachweist. [Sip82; HH86]

Diagonalisierung als Verfahren der Orakelkonstruktion Wie in der Einleitung angesprochen, werden in der Arbeit Orakel konstruiert, relativ zu diesen Implikationen zwischen Pudláks Hypothesen ausgeschlossen werden können. Die Idee der Diagonalisierung eignet sich insbesondere auch, um vollständige Mengen in Komplexitätsklassen auszuschließen, wie beispielsweise die Hypothesen UP und $\mathsf{NP} \cap \mathsf{coNP}$ fordern.

Sei hierfür exemplarisch (M_i) eine Aufzählung an Maschinen von $\mathcal{C}$. Hierbei ist sogar ausreichend, dass die Aufzählung definiert ist, also nicht notwendigerweise aufzählbar im Sinne der Berechenbarkeitstheorie. Sei A die Aussage *Zur Klasse $\mathcal{C}$ existieren keine $\leq_{\mathrm{m}}^{\mathrm{p}}$-vollständigen Sprachen.* Die zu O relativierte Aussage A lautet dann *Zur Klasse $\mathcal{C}^O$ existieren keine $\leq_{\mathrm{m}}^{\mathrm{p},O}$-vollständigen Sprachen.* Es ist zu beachten, dass die Transduktoren der Reduktionen Zugriff auf das Orakel gewonnen haben, und sich der Reduktionsbegriff auf $\leq_{\mathrm{m}}^{\mathrm{p},O}$-Reduktionen verstärkt hat.

Ähnlich wie zu Beginn des Abschnittes reicht es nun aus, für jede Maschine M_i in $\mathcal{C}$ eine Zeugensprache L_i anzugeben, sodass $L_i \not\leq_m^{p,O} L(M_i)$, und $L_i \in \mathcal{C}$. Da sich alle FP^O-Transduktoren aufzählen lassen, ist

$$\forall i, k \in \mathbb{N}\ \exists x : [x \in L(M^O_{\sigma(i)}) \ \Leftrightarrow\ T^O_k(x) \notin L(M^O_i)]$$

eine hinreichende Voraussetzung für A relativ zu O. Hierbei ist $M_{\sigma(i)}$ als die Maschine zu verstehen, welche die Zeugensprache $L_i \in \mathcal{C}$ entscheidet. Damit ist jede Maschine M_i in $\mathcal{C}$ nicht in der Lage, vollständig für $\mathcal{C}$ zu sein.

Die Konstruktion von O lässt sich also durch folgendes Vorgehen beschreiben:

- Wahl einer Definition für die Zeugensprachen L_i abhängig für jede Maschine M_i. Diese muss durch Maschine $M^O_{\sigma(i)}$ entscheidbar sein, und soll dabei das Orakel in ihr Akzeptanzverhalten miteinbeziehen.
- Die eigentliche Diagonalisierung: Für jedes Paar (M_i, T_j) wird ein x gewählt und O so definiert, dass $x \in L(M^O_{\sigma(i)}) \Leftrightarrow T^O_i(x) \notin L(M^O_i)$.

Nichtsdestotrotz ist für die Klassen UP und $\mathrm{NP} \cap \mathrm{coNP}$ eine solche Diagonalisierung ohne Weiteres überhaupt nicht möglich: Für den unrelativierten Fall ließe sich zwar eine Aufzählung an Maschinen bzw. Maschinenpaaren definieren, nicht aber für Orakel-Turing-Maschinen. Hier ist das Akzeptanzverhalten – und damit die Einhaltung des Promises – vom Orakel abhängig.

Ein Ausweg ist beispielsweise für UP die Möglichkeit, zunächst alle Maschinen N_i der Oberklasse $\mathrm{NP}^O \supseteq \mathrm{UP}^O$ zu betrachten. Wir diagonalisieren für jede dieser Maschinen N_i, jeden Transduktor T_k und stellen durch die Wahl von O sicher, dass

(a) die Zeugensprache entscheidende $N^O_{\sigma(i)}$ kategorisch ist, und kein T^O_k realisiert die Reduktion $L(N^O_{\sigma(i)}) \leq_{\mathrm{m}}^{\mathrm{p},O} L(N^O_i)$, oder

(b) für ein x die Maschine $N^O_i(x)$ nicht kategorisch akzeptiert.

Dann kann keine vollständige Sprache mehr existieren: Angenommen, es würde eine solche vollständige Sprache $L \in \mathrm{UP}^O$ existieren, dann existiert kategorische N^O mit $L(N^O) = L$. Ähnlich wie im Beweis von Lemma 3(ii) existiert dann eine äquivalente kategorische Maschine N^O_i mit $L(N^O_i) = L$. Bezüglich dieser Maschine gilt nun oberer Fall (a), also ist $L(N^O_{\sigma(i)}) \not\leq_m^{p,O} L(N^O_i)$, obwohl $N^O_{\sigma(i)}$ kategorisch ist. Damit ist L nicht vollständig; Widerspruch zur Annahme.

Diese Beobachtung erklärt das Leitmotiv der kommenden Diagonalisierungsverfahren: Wir erzielen zunächst, dass (a) die Zeugensprache in der Promise-Klasse liegt, und eine Reduktion auf die Maschinensprache ausgeschlossen ist. Ist das nicht möglich, wird (b) der Promise der jeweiligen Maschineninstanz bewusst zerstört. In beiden Fällen ist dann die Maschinensprache nicht mehr Kandidat für eine vollständige Sprache.

Wir schließen das Kapitel der Grundlagen mit einer Beobachtung über die Reduktionen ab, und zeigen, dass wir uns auf Transduktoren ohne Orakelzugriff beschränken können.

Lemma 5. *Sei O ein beliebiges Orakel.*

(i) Es existieren $\leq^{\mathrm{p}}_{\mathrm{m}}$-vollständige Sprachen für UP^O genau dann wenn $\leq^{\mathrm{p},O}_{\mathrm{m}}$-vollständige Sprachen existieren.

(ii) Es existieren $\leq^{\mathrm{p}}_{\mathrm{m}}$-vollständige Sprachen für $\mathrm{NP}^O \cap \mathrm{coNP}^O$ genau dann wenn $\leq^{\mathrm{p},O}_{\mathrm{m}}$-vollständige Sprachen existieren.

Beweis. *Zu (i):* Zur Implikation von links nach rechts ist die Einsicht ausreichend, dass ohne Beschränkung die Berechnung $T_i(x)$ des Transduktors einer $\leq^{\mathrm{p}}_{\mathrm{m}}$-Reduktion keine Orakelfragen stellt. Damit verhält sich T_i mit Orakel O identisch. Entsprechend realisiert T_i auch die $\leq^{\mathrm{p},O}_{\mathrm{m}}$-Reduktion auf die vollständige Sprache.

Wir zeigen nun die nontriviale Implikation von rechts nach links. Sei also $L \in \mathrm{UP}^O$ die $\leq^{\mathrm{p},O}_{\mathrm{m}}$-vollständige Sprache. Wir definieren

$$L' =_{\mathrm{df}} \{\langle k, 1^{t_k(|x|)}, x\rangle \mid T^O_k(x) \in L\},$$

$L' \in \mathrm{UP}^O$ über folgende Arbeitsweise:

> Simuliere die Berechnung T^O_k über den universellen Transduktor U^O_T. Berechne also $y = U^O_T(k, 1^{t_k(|x|)}, x) = T^O_k(x)$. Bestimme anschließend $y \in L$ und akzeptiere entsprechend.

Die Simulation von $T^O_k(x)$ ist auf polynomielle Laufzeit abhängig zur Eingabelänge beschränkt. Die Entscheidung $T^O_k(x) \in L$ nach Wahl von L durch Ausführen eine kategorische Maschine möglich.

L' ist $\leq^p_m$-hart: Sei $A \in \mathrm{UP}^O$ beliebig, und T^O_k der Transduktor, der die Reduktion $A \leq^{p,O}_m L$ realisiert. Dann ist durch $f\colon x \mapsto \langle k, 1^{t_k(|x|)}, x\rangle$ eine FP-Funktion gegeben, die ohne Zugriff auf O die Reduktion $A \leq^{\mathrm{p}}_{\mathrm{m}} L'$ realisiert:

$$x \in A \;\Leftrightarrow\; T^O_k(x) \in L \;\Leftrightarrow\; \langle k, 1^{t_k(|x|)}, x\rangle \in L' \;\Leftrightarrow\; f(x) \in L'.$$

Zu (ii): Sei $L \in \mathrm{NP}^O \cap \mathrm{coNP}^O$ vollständig. Wir definieren L' analog, der Beweis erfolgt analog zu (i), nur der Promise $L' \in \mathrm{NP}^O \cap \mathrm{coNP}^O$ bleibt zu zeigen.

Nach analoger Laufzeitabschätzung ist auch $L' \in \mathrm{NP}^O$, gleichzeitig ist trotzdem $\overline{L'} \in \mathrm{NP}^O$, denn

$$\langle k, 1^{t_k(|x|)}, x\rangle \in \overline{L'} \;\Leftrightarrow\; T_k^O(x) \notin L \;\Leftrightarrow\; T_k^O(x) \in \overline{L},$$

und nach Wahl von L ist $\overline{L} \in \mathrm{NP}^O$. Die Entscheidung $T_k^O(x) \in \overline{L}$ ist also in NP^O möglich. [vgl. Sip82, S. 524f] □

Für den Rest dieser Arbeit können wir uns also bei den Beweisen auf Reduktionen beschränken, die nicht mit dem Orakel interagieren.

3 Relativ zu einem Orakel sind UP und NP ∩ coNP nicht vollständig

In diesem Kapitel konstruieren wir das Orakel V, relativ zu diesem keine der Klasssen UP und NP ∩ coNP vollständig ist. Wie in der Einleitung bereits angesprochen, separiert dieses Orakel damit weiter Pudláks definierte Hypothesen UP und $\neg\mathsf{NP}\cap\mathsf{coNP}$. Insbesondere ist durch das Orakel gezeigt, dass mit relativierbaren Beweisen die Implikation $\mathsf{UP} \Rightarrow \neg\mathsf{NP}\cap\mathsf{coNP}$ nicht bewiesen werden kann.

Orakel, die für jeweils nur eine der beiden Klassen vollständige Sprachen ausschließen, wurden schon von Sipser für NP ∩ coNP [Sip82], und von Hartmanis und Hemachandra für UP [HH86] bereits konstruiert. Die folgenden zwei Lemmata 8 und 9 isolieren die essenziellen Ideen in den jeweiligen Konstruktionen, über die wir dann V mit den gewünschten Eigenschaften konstruieren können.

Wir definieren zunächst folgende Zeugensprachen:

$$\begin{aligned}
D_j(V) &=_{\text{df}} \{1^n \mid s \geq 1,\ n = p_j^s,\ \ \exists z : |z| = n \wedge z \in V\} \\
X_{a,b}(V) &=_{\text{df}} \{1^n \mid s \geq 1,\ n = p_{a,b}^s, \exists z : |z| = n \wedge z \text{ gerade} \wedge z \in V\} \\
Y_{a,b}(V) &=_{\text{df}} \{1^n \mid s \geq 1,\ n = p_{a,b}^s, \forall z : |z| = n \wedge z \text{ ungerade} \rightarrow z \notin V\}
\end{aligned}$$

Sofort können wir folgende Beobachtung über die Zeugensprachen festhalten:

Beobachtung 6. *Sei V ein beliebiges Orakel.*

(i) $D_j(V), X_{a,b}(V) \in \mathrm{NP}^V$,

(ii) $Y_{a,b}(V) \in \mathrm{coNP}^V$.

(iii) *Wenn für alle Potenzen $n = p_j^s$ auch $|V^{=n}| \leq 1$, dann* $D_j(V) \in \mathrm{UP}^V$.

(iv) *Wenn für alle Potenzen $n = p_{a,b}^s$ auch $1^n \in X_{a,b}(V) \Leftrightarrow 1^n \in Y_{a,b}(V)$, dann* $X_{a,b} \in \mathrm{NP}^V \cap \mathrm{coNP}^V$.

Beweis. *Zu (i):* $D_j(V)$ (bzw. $X_{a,b}(V)$) kann in nichtdeterministischer Polynomialzeit entschieden werden: Auf Eingabe 1^n werden nichtdeterministisch alle (bzw. alle geraden) Zeugen z, $|z| = n$ bestimmt. Die Eingabe wird genau dann angenommen, wenn $z \in V$.

Zu (ii): $\overline{Y_{a,b}(V)}$ kann in nichtdeterministischer Polynomialzeit entschieden werden: Auf Eingabe 1^n (Eingaben anderer Form werden immer akzeptiert) werden nichtdeterministischer alle ungeraden Zeugen z, $|z| = n$ bestimmt. Die Eingabe wird genau dann angenommen, wenn $z \in V$

Zu (iii): Der in (i) angegebene Algorithmus hat höchstens so viele annehmende Pfade, wie verschiedene Zeugen z, $|z| = n$ mit $z \in V$ existieren. Da nach Voraussetzung $|V^{=n}| \leq 1$, existiert nur höchstens ein Rechenweg; also ist der UP-Promise gegeben.

Zu (iv): $X_{a,b}(V) \in \mathrm{NP}^V$ nach Aussage (i). $\overline{X_{a,b}(V)}$ kann in nichtdeterministischer Polynomialzeit entschieden werden: Auf Eingabe 1^n (Eingaben anderer Form werden

immer akzeptiert) wird nichtdeterministisch $1^n \in \overline{Y_{a,b}(V)}$ (nach (ii)) entschieden, und entsprechend die Eingabe akzeptiert. Nach Voraussetzung also genau dann wenn $1^n \in \overline{X_{a,b}(V)}$. □

Folgende kombinatorischen Beobachtung wird uns helfen, die Promises der diskutierten Klassen konstruktiv zu zerstören.

Lemma 7. *Sei $\alpha \in \mathbb{N}$, $\alpha \geq 2$, sei $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ ein gerichteter Graph mit $m = |\mathcal{V}|$, und der Außengrad jedes Knotens ist auf $\leq d$ beschränkt.*

Gilt $m > \alpha(2d+1)$, dann existiert eine unabhängige Teilmenge $\mathcal{U} \subseteq \mathcal{V}$ mit $|\mathcal{U}| = \alpha$, also sodass im induzierten Teilgraph $\mathcal{G}[\mathcal{U}]$ keine verschiedenen Knoten adjazent sind.

Beweis. Sei im Folgenden mit $N(v)$ (bzw. $N^-(v)$, $N^+(v)$) die Nachbarschaftsmenge (bzw. Vorgängermenge, bzw. Nachfolgermenge) von v *im Graph* $\mathcal{G}$ gemeint.

Wir zeigen die Aussage konstruktiv und geben eine Vorschrift an, wie man von einer unabhängigen Menge $\mathcal{U}$ der Größe i auf eine der Größe $i+1 \leq \alpha$ schließen kann. Damit haben wir am Ende die gesuchte unabhängige Teilmenge der Größe α konstruiert. Wir sichern dabei immer, dass für alle $u \in \mathcal{U}$ der Innengrad auf $\leq d$ beschränkt ist, das heißt $|N^-(u)| \leq d$ $(*)$.

(Basisfall.) Ist $i = 0$, dann ist durch $\mathcal{U} = \emptyset$ offenbar eine Teilmenge gegeben, für die $\mathcal{G}[\mathcal{U}]$ keine Kanten enthält.

(Rekursionsschritt.) Sei $\mathcal{U}$ eine unabhängige Menge mit $i =_{\text{df}} |\mathcal{U}| < \alpha$. Zunächst entfernen wir $\mathcal{U}$ und die Nachbarschaft $N(\mathcal{U}) = N^+(\mathcal{U}) \cup N^-(\mathcal{U})$ aus $\mathcal{G}$. Sei also dann $\mathcal{G}' = \mathcal{G}[\mathcal{V} \setminus \mathcal{U} \setminus N(\mathcal{U})] = (\mathcal{V}', \mathcal{E}')$. Insbesondere ist mit $(*)$ dann $|N^-(u)|, |N^+(u)| \leq d$ für alle $u \in \mathcal{U}$. Damit verbleiben

$$\begin{aligned} |\mathcal{V}'| &= |\mathcal{V}| - |\mathcal{U}| - |N(\mathcal{U})| \geq m - i - |N^+(\mathcal{U})| - |N^-(\mathcal{U})| \\ &\geq m - i(2d+1) > m - \alpha(2d+1) > 0 \end{aligned}$$

viele Knoten in $\mathcal{G}'$. Wähle nun in $\mathcal{G}'$ einen Knoten v mit $\text{indeg}_{\mathcal{G}'}(v) \leq d$. Dieser Knoten existiert, ansonsten wäre widersprüchlicherweise

$$|\mathcal{E}'| = \sum_{v \in \mathcal{V}'} \text{indeg}_{\mathcal{G}'}(v) > d|\mathcal{V}'| \geq \sum_{v \in \mathcal{V}'} \text{outdeg}(v) \geq \sum_{v \in \mathcal{V}'} \text{outdeg}_{\mathcal{G}'}(v) = |\mathcal{E}'|.$$

Die Menge $\mathcal{U}' =_{\text{df}} \mathcal{U} \cup \{v\}$ erfüllt die geforderten Eigenschaften:

Die neue Menge $\mathcal{U}'$ ist auf $i+1$ angewachsen, ist ja $v \notin \mathcal{U}$. Sie ist unabhängig, ist ja $v \notin N(\mathcal{U})$. Weiterhin ist $(*)$ gesichert: Da insbesondere $v \notin N^+(\mathcal{U})$ ist $N^-(v) \subseteq \mathcal{V}'$. Nach Wahl von v ist $\text{indeg}_{\mathcal{G}'}(v) \leq \text{indeg}(v) \leq d$, damit also $|N^-(v)| \leq d$. □

Anmerkung. Bis auf wenige Einzelfälle wird die Aussage des Lemmas nur mit Wahl $\alpha = 2$ angewandt. Hier reduziert sich dann die Ungleichung in der Voraussetzung zu $m - 2 > 4d$.

Lemma 8. *Sei $j, k \in \mathbb{N}$ beliebig, $n = p_j^s$ hinreichend groß sodass $2^n - 2 > 4\, t_j(t_k(n))$, und $C \subseteq \Sigma^*$ mit $|c| < n$ für alle $c \in C$.*

Es existiert eine Erweiterung $E \subseteq \Sigma^n$ sodass einer der folgenden Fälle eintritt:

(i) $|E| \leq 1$ *mit* $1^n \in D_j(C \cup E) \Leftrightarrow T_k(1^n) \notin L(N_j^{C \cup E})$.

(ii) $|E| = 2$ *und* $N_j^{C \cup E}$ *akzeptiert* $T_k(1^n)$ *auf zwei Rechenwegen.*

Beweis. Sei im Folgenden $y =_{\text{df}} T_j(1^n)$. Angenommen, keine Erweiterung E existiert sodass Fall (i) oder Fall (ii) eintritt. Dann ist $y \notin L(N_j^C)$, und für alle $w \in \Sigma^n$ auch $y \in L(N_j^{C \cup \{w\}})$. Mittels Lemma 7 zeigen wir, dass dann widersprüchlicherweise Fall (ii) erreicht werden kann.

Für alle $w \in \Sigma^n$ definieren wir $Q(w)$ als die Menge aller Orakelfragen auf dem (lexikalisch kleinsten) Annahmepfad von $N_j^{C \cup \{w\}}(y)$. Insbesondere ist $w \in Q(w)$, wäre ansonsten entsprechender Rechenweg auf Berechnung $L(N_j^C)$ vorhanden. Dann wäre y akzeptiert worden, und Fall (i) widersprüchlicherweise erfüllt.

Wir definieren den Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ mit

$$\mathcal{V} =_{\text{df}} \Sigma^n, \quad \mathcal{E} =_{\text{df}} \{(u, v) \mid v \in Q(u)\}.$$

Es ist $|\mathcal{V}| = 2^n =_{\text{df}} m$, und der Außengrad jedes Knotens u auf $|Q(u)| \leq t_j(|y|) \leq t_j(t_k(n)) =_{\text{df}} d$ beschränkt. Sei $\alpha =_{\text{df}} 2$, und nach Wahl von n ist $m > \alpha(2d + 1)$, und Voraussetzungen des Lemmas erfüllt, und es existiert eine zwei-elementige unabhängige Menge $\{u, v\} \subseteq \Sigma^n$, also mit $v \notin Q(u)$, $u \notin Q(v)$.

Sei dann $E =_{\text{df}} \{u, v\}$. Nach Annahme wird $N_j^{C \cup \{u\}}(y)$ und $N_j^{C \cup \{v\}}(y)$ akzeptieren. Die (jeweils lexikalisch kleinsten) akzeptierenden Rechenwege sind paarweise verschieden, ist ja $u \in Q(u)$, aber $u \notin Q(v)$.

Da $v \notin Q(u)$ wird v nicht auf dem entsprechenden Rechenweg gefragt, also akzeptiert auch $N_j^{C \cup E}(y)$ auf dem identischen Rechenweg. Analog für den Rechenweg von $Q(v)$, damit akzeptiert $N_j^{C \cup E}(y)$ auf zwei verschiedenen Rechenwegen. Fall (ii) tritt ein, Widerspruch zur Annahme. [vgl. HH86, Thm. 3.1, Lem. 3.2] □

Lemma 9. *Seien $a, b, k \in \mathbb{N}$ beliebig, q ein Polynom was die Laufzeit von N_a und N_b beschränkt, $n = p_{a,b}^s$ hinreichend groß sodass $2^n - 2 > 4\, q(t_k(n))$, und $C \subseteq \Sigma^*$ mit $|c| < n$ für alle $c \in C$.*

Es existiert eine Erweiterung $E \subseteq \Sigma^n$ sodass einer der folgenden Fälle eintritt:

(i) Die Maschinen $N_a^{C \cup E}$, $N_b^{C \cup E}$ arbeiten komplementär, und

$$1^n \in Y_{a,b}(C \cup E) \;\Leftrightarrow\; 1^n \in X_{a,b}(C \cup E) \;\Leftrightarrow\; T_k(1^n) \notin L(N_a^{C \cup E}).$$

(ii) $N_a^{C \cup E}$ und $N_b^{C \cup E}$ arbeiten nicht komplementär.

Beweis. Sei im Folgenden $y =_{\text{df}} T_j(1^n)$. Angenommen, keine Erweiterung E existiert sodass Fall (i) oder Fall (ii) eintritt. Mittels Lemma 7 zeigen wir, dass dann widersprüchlicherweise Fall (ii) erreicht werden kann.

Für E mit $|E| = 1$ ist offenbar $1^n \in Y_{a,b}(C \cup E) \Leftrightarrow 1^n \in X_{a,b}(C \cup E)$, ist ja das einzige Wort in E entweder gerade oder ungerade. Da (i) nicht eintritt, wird deshalb für alle geraden $u \in \Sigma^n$ die Berechnung $N_a^{C\cup\{u\}}(y)$ akzeptieren, für alle ungeraden $v \in \Sigma^n$ wird die Berechnung $N_a^{C\cup\{v\}}(y)$ ablehnen. Und da (ii) nicht eintritt, wird also $N_b^{C\cup\{v\}}(y)$ akzeptieren.

Für alle geraden $u \in \Sigma^n$ definieren wir $Q_a(u)$ als die Menge aller Orakelfragen, die auf dem (lexikalisch kleinsten) Annahmepfad von $N_a^{C\cup\{u\}}(y)$ gestellt werden. Analog definieren wir $Q_b(v)$ für alle ungeraden $v \in \Sigma^n$.

Wieder ähnlich zum vorigen Beweis definieren wir einen Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ mit

$$\mathcal{V} =_{\text{df}} \Sigma^n, \quad \mathcal{E} =_{\text{df}} \{(u,z) \mid u \text{ gerade}, z \in Q_a(u)\} \cup \{(v,z) \mid v \text{ ungerade}, z \in Q_b(v)\}.$$

Es ist $|\mathcal{V}| = 2^n =_{\text{df}} m$, und der Außengrad jedes Knotens auf $\leq q(|y|) \leq q(t_k(n)) =_{\text{df}} d$ beschränkt. Sei $\alpha =_{\text{df}} 2$, nach Wahl von n ist $m > \alpha(2d+1)$, und Voraussetzungen des Lemma 7 erfüllt. Es existieren $u, v \in \Sigma^n$, u gerade, v ungerade, $v \notin Q_a(u)$, $u \notin Q_b(v)$.

Sei dann $E =_{\text{df}} \{u, v\}$. Wir zeigten bereits, dass $N_a^{C\cup\{u\}}(y)$ akzeptiert. Da $v \notin Q_a(u)$, wird v nicht in der akzeptierenden Berechnung gefragt, also akzeptiert auch $N_a^{C\cup E}(y)$. Analog wird dann auch $N_b^{C\cup E}(y)$ akzeptieren. Damit ist Fall (ii) eingetreten, Widerspruch zur Annahme. [vgl. Sip82, S. 525ff; vgl. DG19, Lemma 2.14] □

Wir konstruieren nun das angekündigte Orakel V, indem mit den eben gezeigten Lemmata für alle FP-Funktionen über sowohl UP-Maschinen, als auch über NP∩coNP-Maschinenpaare diagonalisiert wird. Dabei bauen wir das Orakel stufenweise auf, und gehen so vor wie im vorigen Kapitel skizziert. In jeder Stufe erzielen wir ein neues ›Fehlverhalten‹, welches in höheren Stufen weiterhin gilt.

Satz 10. *Es existiert ein Orakel V sodass*

(i) keine $\leq_m^p$-vollständige Sprache für UP^V existiert, und

(ii) keine $\leq_m^p$-vollständige Sprache für $\text{NP}^V \cap \text{coNP}^V$ existiert.

Beweis. Wir definieren $V =_{\text{df}} \bigcup_{i=0}^{\infty} V_i$ und sichern $V_0 \subseteq V_1 \subseteq V_2 \subseteq \cdots$. Sei

$$\mathcal{T} =_{\text{df}} \{t_{a,b,k} \mid a, b, k \in \mathbb{N}, a \neq b\} \cup \{t_{j,k} \mid j, k \in \mathbb{N}\}$$

eine Liste an Tasks, und seien durch $t_{(1)}, t_{(2)}, \ldots$ alle Tasks aufgezählt. In jeder Stufe i der Konstruktion werden wir dabei den Task $t_{(i)}$ erfüllen. Wir starten mit $V_0 =_{\text{df}} \emptyset$, $n_0 =_{\text{df}} 0$. Folgende Vorschrift gibt dann induktiv die Konstruktion für Stufe $i > 0$ an.

(1) Ist $t_{(i)} = t_{j,k}$, führen wir eine Diagonalisierung über die UP-Maschine N_j durch. Wähle $n = p_j^s > n_{i-1}$ sodass $2^n - 2 > 4\,t_j(t_k(n))$. Nach Lemma 8 existiert zu V_{i-1} eine Erweiterung $V_i =_{\text{df}} V_{i-1} \cup E$ sodass

   (a) entweder $1^n \in D_j(V_i) \Leftrightarrow T_k(1^n) \notin L(N_j^{V_i})$ mit höchstens einem Wort der Länge n in V_i,

   (b) oder $N_j^{V_i}$ arbeitet nicht mehr kategorisch.

   Setze $n_i > t_j(t_k(n))$ und gehe zur nächsten Stufe über.

(2) Ist $t_{(i)} = t_{a,b,k}$, führen wir in dieser Stufe eine Diagonalisierung über das NP∩coNP-Paar (N_a, N_b) durch. Seien die beiden Maschinen auf q zeitbeschränkt, das heißt $q(n) =_{\text{df}} n^c + c$ mit $c =_{\text{df}} \max\{a, b\}$. Wähle $n = p_{a,b}^s > n_{i-1}$ sodass $2^n - 2 > 4\,q(t_k(n))$. Nach Lemma 9 existiert zu V_{i-1} eine Erweiterung $V_i =_{\text{df}} V_{i-1} \cup E$ sodass

   (a) entweder $1^n \in Y_{a,b}(V_i) \Leftrightarrow 1^n \in X_{a,b}(V_i) \Leftrightarrow T_k(1^n) \notin L(N_a^{V_i})$,

   (b) oder $N_a^{V_i}$ und $N_b^{V_i}$ arbeiten nicht komplementär.

   Setze $n_i > q(t_k(n))$ und gehe zur nächsten Stufe über.

Zunächst zeigen wir, dass die in den einzelnen Stufen gesicherten ›Fehlverhalten‹ tatsächlich auch im abgeschlossenen Orakel gelten.

Behauptung 11. *In gesicherten Aussagen (1a,1b,2a,2b) kann V_i zu V ersetzt werden, ohne dass diese nicht mehr erfüllt sind.*

Beweis. Wir zeigen die Behauptung nur für die Aussage (1a), der Beweis für die anderen Aussagen erfolgt analog. Sei also $t_{(i)} = t_{j,k}$, $N_j^{V_i}$ kategorisch,

$$1^n \in D_j(V_i) \Leftrightarrow T_k(1^n) \notin L(N_j^{V_i}), \quad |V_i^n| \leq 1.$$

Es ist $1^n \in D_j(V_i) \Leftrightarrow 1^n \in D_j(V)$: Einerseits ist $D_j(V_i) \subseteq D_j(V)$, andererseits wird nach Stufe i kein Wort der Länge n dem Orakel hinzugefügt, damit $1^n \notin D_j(V_i) \Rightarrow 1^n \notin D_j(V)$. Des Weiteren existiert dann auch nur höchstens ein Wort der Länge n in V.

Es ist $T_k(1^n) \in L(N_j^{V_i}) \Leftrightarrow T_k(1^n) \in L(N_j^V)$: Wir beobachten, dass in höheren Stufen nur Wörter der Länge $> t_j(t_k(n))$ hinzugefügt werden. Gleichzeitig ist die Laufzeit von $N_j(T_k(1^n))$ unabhängig vom Orakel auf $\leq t_j(|T_k(1^n)|) \leq t_k(t_k(n))$ beschränkt. Die von N_j gestellten Fragen werden also von V und V_i gleich beantwortet.

Also ist auch N_j^V kategorisch, $|V^{=n}| \leq 1$,

$$1^n \in D_j(V) \Leftrightarrow 1^n \in D_j(V_i) \Leftrightarrow T_k(1^n) \notin L(N_j^{V_i}) \Leftrightarrow T_k(1^n) \notin L(N_j^V). \qquad \square$$

Wir zeigen nun, wie sich die Zeugensprachen nicht auf Promise-erfüllenden Maschinen (bzw. Maschinenpaare) reduzieren lassen kann.

Behauptung 12. *Seien $a, b, j, k \in \mathbb{N}$ beliebig.*

(i) Für kategorische N_j^V realisiert Transduktor T_k nicht die Reduktion $D_j(V) \leq_{\mathrm{m}}^{\mathrm{p}} L(N_j^V)$.

(ii) Für komplementäre N_a^V, N_b^V realisiert Transduktor T_k nicht die Reduktion $X_{a,b}(V) \leq_{\mathrm{m}}^{\mathrm{p}} L(N_a^V)$.

Beweis. Zu (i): Wir führen die Negation zum Widerspruch und nehmen an, dass N_j^V kategorisch ist, und $D_j(V) \leq_{\mathrm{m}}^{\mathrm{p}} L(N_j^V)$ über die von T_k berechnete Reduktionsfunktion.

Nach Voraussetzung ist N_j^V kategorisch, aus Kontraposition von voriger Behauptung 11 folgt dann auch die Negation von (1b) für alle i, das heißt auch für alle i arbeitet $N_j^{V_i}$ kategorisch. Also tritt in Stufe i mit $t_{(i)} = t_{j,k}$ auch Fall (1a) ein. Da, wieder durch vorige Behauptung, die Aussagen für V gesichert sind, gilt für geeignetes n die Aussage

$$1^n \in D_j(V) \;\Leftrightarrow\; T_k(1^n) \notin L(N_j^V).$$

Also realisiert T_k nicht die Reduktion; Widerspruch zur Annahme, also $D_j(V) \not\leq_m^p L(N_j^V)$.

Zu (ii): Analog führen wir die Negation zum Widerspruch und nehmen an, dass T_k die Reduktion $X_{a,b} \leq_{\mathrm{m}}^{\mathrm{p}} L(N_a^V)$ realisiert.

Analog zu (i) tritt in Stufen i mit $t_{(i)} = t_{j,k}$ auch Fall (2a) ein, und für geeignetes n ist dann

$$1^n \in X_{a,b}(V) \;\Leftrightarrow\; T_k(1^n) \notin L(N_a^V).$$

Wieder analog Widerspruch zur Annahme, also $X_{a,b}(V) \not\leq_m^p L(N_a^V)$. □

In der letzten Behauptung vor Abschluss des Beweises zeigen wir, dass in relevanten Fällen die Zeugensprachen tatsächlich auch in den entsprechenden Klassen liegt.

Behauptung 13. *(i) Ist N_j^V eine kategorische Maschine, dann ist $D_j \in \mathrm{UP}^V$.*

(ii) Arbeiten die Maschinen N_a^V und N_b^V komplementär, dann ist $X_{a,b} \in \mathrm{NP}^V \cap \mathrm{coNP}^V$.

Beweis. Zu (i): Angenommen, es existiert ein s sodass mit $n = p_j^s$ auch $|V^{=n}| > 1$. Wir zeigen, dass in keinen Fällen der Konstruktion entsprechende Wörter hinzugefügt werden:

Nicht in (2a) oder (2b), da $n \neq p_{a,b}^{s'}$ für alle $a, b, s' \in \mathbb{N}$. Nicht in (1a), ist nach Behauptung 11 höchstens ein Wort der Länge n in V, also widersprüchlich zur Annahme. Nicht in (1b), ist nach gleicher Behauptung dann N_j^V nicht kategorisch, was sich mit der Voraussetzung widerspricht.

Also ist für alle s auch $|V^{=n}| \leq 1$ mit $n = p_j^s$. Nach Beobachtung 6(iii) dann $D_j \in \mathrm{UP}^V$.

Zu (ii): Wir gehen analog vor und nehmen an, dass ein s existiert sodass mit $n = p_{a,b}^s$ auch $1^n \in X_{a,b}(V) \Leftrightarrow 1^n \notin Y_{a,b}(V)$, und zeigen dass in keinen Fällen der Konstruktion entsprechende Wörter hinzugefügt werden.

Aus analoger Argumentation der Primzahlen nicht in (1a) oder (1b). Ebenso widerspricht sich (2a) bzw. (2b) mit der Annahme, bzw. der Voraussetzung.

Voraussetzung der Beobachtung 6(iv) erfüllt, also $A_{a,b} \in \mathrm{NP}^V \cap \mathrm{coNP}^V$. □

Behauptung 14. *(i) Es existiert keine $\leq_m^p$-vollständige Sprache für UP^V.*

(ii) Es existiert keine $\leq_m^p$-vollständige Sprache für $\mathrm{NP}^V \cap \mathrm{coNP}^V$.

Beweis. *Zu (i):* Angenommen, es existiert eine vollständige Sprache L für UP^V. Sei N die kategorische Maschine, sodass $L = L(N^V)$. Ähnlich wie im Beweis von Lemma 3(ii) lässt sich durch Hinzufügen neuer Zustände zu N eine äquivalente Maschine der Standard-Aufzählung N_j^V angeben, mit $L(N_i^V) = L$. Insbesondere ist N_j^V kategorisch. Nach Behauptung 13(i) ist deshalb $D_j \in \mathrm{UP}^V$, aufgrund Vollständigkeit von L ist dann $D_j \leq_{\mathrm{m}}^{\mathrm{p}} L(N_j^V)$ über ein Transduktor T_k.

Widerspruch zur Behauptung 12(i), T_k realisiert keine solche Reduktion. Annahme war falsch, für UP^V existiert keine vollständige Sprache.

Zu (ii): Wieder nehmen wir an, dass L vollständig für $\mathrm{NP}^V \cap \mathrm{coNP}^V$ ist. Da $L, \overline{L} \in \mathrm{NP}$, existieren nach Lemma 3(ii) a und b sodass $L = L(N_a^V)$, $\overline{L} = L(N_b^V)$, sowie N_a und N_b komplementär. Nach Behauptung 13(ii) ist $X_{a,b} \in \mathrm{NP}^V \cap \mathrm{coNP}^V$, aufgrund Vollständigkeit von L ist dann $X_{a,b} \leq_{\mathrm{m}}^{\mathrm{p}} L(N_a^V)$ über ein Transduktor T_k.

Widerspruch zur Behauptung 12(ii), Annahme war falsch, für UP^V existiert keine vollständige Sprache. □

Die geforderten Eigenschaften von V werden in voriger Behauptung nachgewiesen. Das schließt den Beweis von Satz 10 ab. □

4 Relativ zu einem Orakel ist $\mathsf{P} = \mathsf{NP} \cap \mathsf{coNP}$, aber UP ohne vollständige Sprachen

Im Vergleich zum gerade konstruierten Orakel V wird in diesem Kapitel nun eine Konstruktion eines Orakels W vorgestellt, relativ zu dieser

- zwar genauso wie V die Menge UP keine vollständige Menge besitzt,
- aber, im Unterschied zu V, fordern wir $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$. Damit existieren insbesondere vollständige Sprachen für die Menge $\mathrm{NP} \cap \mathrm{coNP}$.

Insbesondere separiert damit W die Hypothesen UP und $\mathsf{NP} \cap \mathsf{coNP}$. Das Orakel zeigt, dass mit relativierbaren Methoden die Implikation $\mathsf{UP} \Rightarrow \mathsf{NP} \cap \mathsf{coNP}$ nicht bewiesen werden kann.

Das Orakel V vom vorigen Kapitel ließ sich durch *direkte* Diagonalisierung konstruieren. In jeder Stufe der Konstruktion ließ sich immer eine der Vorgaben aus $\mathcal{T}$ endgültig sicherstellen. Im Gegensatz zur Konstruktion des vorigen Kapitels erfordert die zweite Vorgabe $\mathrm{P}^W = \mathrm{NP}^W \cap \mathrm{coNP}^W$ mehr Beachtung. Zunächst ist unklar, wie eine allgemeine Rechenvorschrift möglich ist, die für alle komplementär akzeptierende Maschinenpaare deren Akzeptanz deterministisch bestimmt.

In ihrer Arbeit »Relativizations of the P=? NP Question« [BGS75] beantworten die Autoren Baker, Gill und Solovay eine verwandet Frage: Sie konstruieren ein Orakel, relativ zu diesem $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP} \neq \mathrm{NP}$ gilt. Durch ein geschicktes Verfahren können sie einerseits $\mathrm{P} \neq \mathrm{NP}$ sichern, gleichzeitig einen geeigneten Algorithmus angeben, sodass die anspruchsvollere Eigenschaft $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ besteht.

Damit eignet sich das von Baker et al. vorgestellte Verfahren auch, um das Orakel W mit den oben genannten Eigenschaften zu konstruieren. Anstatt $\mathrm{P} \neq \mathrm{NP}$ sichern wir wie gewünscht die Hypothese UP. Im Rest des Kapitels wird zunächst das Verfahren von Baker et al. informell vorgestellt und zusammengefasst, als da deren Technik die essenzielle Grundlage der anschließenden Konstruktion von W veranschaulicht. Am Ende des Kapitels wird dann die formale Konstruktion von W angegeben.

4.1 Grundlage: Bisherige Verfahren für $\mathsf{P} = \mathsf{NP} \cap \mathsf{coNP} \neq \mathsf{NP}$

Wie angesprochen, orientiert sich die Konstruktion von W an einem Verfahren von Baker, Gill und Solovay, mit welchem ein Orakel B konstruiert werden konnte, relativ zu diesem $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP} \neq \mathrm{NP}$ gilt. [BGS75, Theorem 6, dort lautet das Orakel E]

Baker et al. starten dabei mit einem beliebigen Orakel A sodass $\mathrm{P}^A = \mathrm{NP}^A$. Schon eine für PSPACE vollständige Menge wie `QBF` – die Menge der erfüllbaren quantifizierte booleschen Formeln ohne freie Variablen – ist als Wahl für A günstig; auch das zeigten schon Baker et al. [BGS75, S. 434]

In Anwesenheit von A lässt sich also schon in deterministischer Polynomialzeit entscheiden, ob eine nichtdeterministisch polynomiell beschränkte Berechnung akzeptiert.

Spezieller sollte sich dann auch jede nichtdeterministische Maschine N_i^A durch eine deterministische Maschine $P_{i'}^A$ simulieren lassen können. Durch die deterministisch durchgeführte Simulation von $N_i^A(x)$ sollte demnach intuitiv auch möglich sein, einen akzeptierenden Rechenweg in der Berechnung von $N_i^A(x)$ zu bestimmen (sollte ein solcher Rechenweg überhaupt existieren). Diese Beobachtung sichern wir im folgenden Lemma, bevor mit der Beschreibung des Verfahrens fortgeführt wird.

Lemma 15. *Sei A so gewählt dass $\mathrm{P}^A = \mathrm{NP}^A$. Zu jeder nichtdeterministischen Maschine N_i^A lässt sich über Zugriff auf das Orakel A in deterministischer Polynomialzeit bestimmen, ob N_i^A die Eingabe x akzeptiert, und falls ja, mit welchem Rechenweg $N_i^A(x)$ akzeptiert.*

Insbesondere existiert $\mathtt{AKZ} \in \mathrm{FP}^A$ mit folgendem Verhalten:

- $\mathtt{AKZ}(i, 1^{t_i(|x|)}, x) = \varepsilon$, *wenn $N_i^A(x)$ ablehnt.*
- $\mathtt{AKZ}(i, 1^{t_i(|x|)}, x) = \langle \alpha_1, \ldots, \alpha_m \rangle$, *wenn $N_i^A(x)$ akzeptiert. Dabei bezeichnet die endliche Folge von Konfigurationen $\alpha_1, \ldots, \alpha_m$ einen akzeptierenden Rechenweg von $N^A(x)$.*

Beweis. Unter *Konfigurationen* ist eine Codierung des Zustandes sowie der Bandinhalte und Kopfpositionen gemeint. Offenbar ist die Größe einer solchen Konfiguration von Maschinen N_i auf $\mathcal{O}(t_i(|x|))$ beschränkt.

Wir definieren

$$\begin{aligned}\mathtt{INIT} =_{\mathrm{df}} \{\langle i, x, 1^{t_i(|x|)}, \alpha_1, \ldots, \alpha_k \rangle \mid{}& \text{es existieren } \alpha_{k+1}, \ldots, \alpha_m,\ m \leq t_i(|x|) \\ & \text{sodass } \alpha_1, \ldots, \alpha_m \text{ ein akzeptierender} \\ & \text{Rechenweg von } N_i^A(x) \text{ ist}\}.\end{aligned}$$

Es ist $\mathtt{INIT} \in \mathrm{NP}^A$: Sei hierfür n die Länge von $\langle i, x, \ldots \rangle$. Nichtdeterministisch wird zunächst $m \leq t_i(|x|) \leq n$ bestimmt. Dann lässt sich nichtdeterministisch in Zeit $\mathcal{O}(m \cdot t_i(|x|)) = \mathcal{O}(n^2)$ eine Folge von Konfigurationen $\alpha_{k+1}, \ldots, \alpha_m$ wählen. Ob $\alpha_1, \ldots, \alpha_m$ ein gültigen und akzeptierenden Rechenweg darstellt, lässt sich anhand der Übergangsfunktion (und Zugriff auf A) in deterministischer Polynomialzeit abhängig von n prüfen.

Nach Wahl von A ist dann auch $\mathtt{INIT} \in \mathrm{P}^A$. Über diese Beobachtung lässt sich nun ein Algorithmus für $\mathtt{AKZ}$ auf Eingabe $\langle i, 1^{t_i(|x|)}, x \rangle$ mit Gesamtlänge n mit folgender Vorschrift beschreiben:

(1) Bestimme die Startkonfiguration α_1 der Maschine N_i. Setze $k \leftarrow 1$.

(2) Ist $\langle i, x, 1^{t_i(|x|)}, \alpha_1 \rangle \notin \mathtt{INIT}$, terminiere mit Ausgabe ε. Andernfalls gehe zum nächsten Schritt. Insbesondere existiert dann ein akzeptierender Rechenweg für $N_i^A(x)$.

(3) Solange α_k keine akzeptierende Endkonfiguration ist: Iteriere über alle von α_k in einem Schritt erreichbaren Konfigurationen α_{k+1}. Diese Anzahl erreichbarer Konfigurationen ist hierbei durch die Übergangsrelation von N_i auf $O(|i|)$ beschränkt.

(4) Ist $\langle i, x, 1^{t_i(|x|)}, \alpha_1, \ldots, \alpha_k, \alpha_{k+1}\rangle \in \texttt{INIT}$, speichere α_{k+1} und setze $k \leftarrow k+1$. Springe anschließend zurück zu Schritt (2).

(5) Terminiere mit Ausgabe $\langle \alpha_1, \ldots, \alpha_k\rangle$.

Mit Zugriff auf A läuft der Algorithmus in deterministischer Polynomialzeit abhängig n: Schritt (2) und (4) lassen sich deterministisch durchführen, ist ja $\texttt{INIT} \in \mathrm{P}^A$. Weiterhin sind aufgrund der Laufzeitschranke von N_i auch nur $\leq t_i(|x|)$, also linear viele Iterationen abhängig von n möglich. Also $\texttt{AKZ} \in \mathrm{FP}^A$. [vgl. BGS75, Lemma 2] □

Durchführung der Konstruktion Essenzieller Bestandteil der Konstruktion ist die Idee, sehr weit auseinander liegende ›Lücken‹ auf festen Stufen in dem anfänglichen Orakel A zu fordern. Einerseits lassen die Lücken Freiheitsgrade zu, sodass eine Diagonalisierung wie in dem von ihnen vorgestellten P≠NP-Orakel [BGS75, Thm. 3] erfolgen kann. Dabei werden die Lücken mit höchstens einem Wort gefüllt.

Könnten zwei komplementär akzeptierende nichtdeterministischen Maschinen schnell *deterministisch* die Inhalte der Lücken, also $B \setminus A$, bestimmen, so wäre über A dann deterministisch möglich zu entscheiden, welche NP-Maschine die Eingabe akzeptiert.

Zwar lässt sich nicht die gesamte Menge $B \setminus A$ entscheiden, andererseits liegen die Lücken an festen Stufen und sehr weit auseinander. So können die zwei komplementär akzeptierende Maschinen über geschickte Weise kombiniert werden, um zumindest eine endlich große relevante Portion von $B \setminus A$ deterministisch zu bestimmen. Diese ermöglicht dann $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$.

Konkret definieren die Autoren

$$e(0) =_{\mathrm{df}} 1, \quad e(i+1) =_{\mathrm{df}} 2^{(2^{e(i)})},$$

wobei keine Wörter der Länge $e(\cdot)$ in A existieren sollen; dort stehen dann die Lücken, die durch die Diagonalisierung besetzt werden. Ein solches A können wir beispielsweise folgendermaßen definieren

$$A =_{\mathrm{df}} \{0^n 1 q \mid q \in \texttt{QBF} \wedge \forall i : |0^n 1 q| \neq e(i)\}.$$

Hier bezeichnet $q \in \Sigma^*$ also die Codierung einer quantifizierten booleschen Formel. Offenbar ist für alle i auch $A \cap \Sigma^{e(i)} = \emptyset$, sowie A vollständig für PSPACE. Damit $\mathrm{P}^A = \mathrm{NP}^A$ wie gefordert. Die Definitionen von $e(\cdot)$ und A verwenden wir dabei auch für den Rest dieser Arbeit.

In der Konstruktion wird die P≠NP-Bedingung dann über direkte Diagonalisierung realisiert: Ab hinreichend großer Stufe i kann eine deterministische Berechnung $P(1^{e(i)})$ nicht alle Wörter der Länge $e(i)$ des Orakels erfragen. Ein solches nicht erfragtes Wort wird dann geeignet in die Lücke $e(i)$ eingesetzt, sodass $L(P)$ nicht die NP-Zeugensprache $\{1^{e(i)} \mid B \cap \Sigma^{e(i)} \neq \emptyset\}$ entscheidet.

Gleichzeit dazu wird – vereinfacht ausgesprochen – versucht, bei möglichst allen Maschinenpaaren den NP∩coNP-Promise zum frühstmöglichen Zeitpunkt während der Konstruktion zu zerstören. Wird also $L \in \mathrm{NP}^B \cap \mathrm{coNP}^B$ durch das Maschinenpaar $L = L(N_a^B), \overline{L} = L(N_b^B)$ entschieden, ließ sich der Promise des Paars zu keinem Zeitpunkt der Konstruktion zerstören; N_a und N_b akzeptieren gewissermaßen ›inhärent‹ komplementär.

Deterministisches Bestimmen relevanter Orakelwörter Mithilfe des inhärent komplementär akzeptierenden Paars lässt sich nun eine Möglichkeit angeben, um $x \in L$ deterministisch zu entscheiden. Unser Ziel ist es, die Inhalte der Lücken deterministisch zu berechnen. Beantworten wir dann in der Berechnung von N_a^B die Orakelfragen der Länge $e(\cdot)$ durch eine endliche Menge, können wir B durch A in der Berechnung austauschen. Und da $\mathrm{NP}^A = \mathrm{P}^A$, können wir abschließend deterministisch entscheiden, ob N_a^B akzeptiert, also ob $x \in L$ liegt.

Es bleibt noch offen, wie wir die Orakelwörter der Länge $e(\cdot)$ bestimmen. Ab hinreichend langem Wort x existiert genau ein i sodass

$$e(i-1) < \log|x| \leq e(i), \text{ und } \max\{t_a(|x|), t_b(|x|)\} < e(i+1).$$

Insbesondere lässt sich dann die Lücken $e(0), e(1), \ldots, e(i-1)$ vollständig erfragen und die Menge

$$B' =_{\mathrm{df}} \{w \mid w \in B, |w| = e(i'), i' < i\}$$

bestimmen. Auf der Berechnung von $N_a^B(x)$ können trotzdem noch Fragen aus der Lücke $e(i)$ gestellt werden, aber aus keiner höheren Lücke, ist ja die Laufzeit auf $< e(i+1)$ beschränkt. Es reicht also aus, das (höchstens einzige) Wort aus $B^{=e(i)}$ zu finden.

Hier ist der ›inhärente‹ Promise des Maschinenpaares behilflich: Die Konstruktion stellte sicher, dass auch $N_a^{A\cup B'}(x)$ und $N_b^{A\cup B'}(x)$ komplementär akzeptieren. (Ansonsten hätte die Konstruktion die Lücke $e(i)$ leer gehalten, und so das Promise des Paares zerstört.) Im Allgemeinen aber wird das Akzeptierverhalten der beiden Maschinen zwischen $A \cup B'$ und B tauschen.

Analog wie eben beschrieben ersetzen wir Orakelfragen der Länge $e(\cdot)$ durch Fragen an die Menge B', und können B durch A austauschen. Wir bestimmen damit den akzeptierenden Pfad von $N_a^{A\cup B'}$ oder $N_b^{A\cup B'}$ über `AKZ` aus Lemma 15. Aus diesem Pfad können wir nun die gestellten Orakelfragen der Länge $e(i)$ ablesen. Es ergeben sich zwei Fälle:

(a) Keine dieser Orakelfragen liegt in B. Dann wird sich N_a und N_b offenbar mit $A\cup B'$ nicht anders als mit B verhalten. Die Maschine mit dem akzeptierenden Pfad ist dann also auch die Maschine, die mit B akzeptiert. Damit dann klar entscheidbar, ob $x \in L$.

(b) Oder eine Orakelfrage w der Länge $e(i)$ liegt in B und wurde in der Simulation fälschlicherweise negativ beantwortet. Dann haben wir das einzige $w \in B$ dieser Länge

auch gefunden, und nach voriger Argumentation ist das ausreichend, um $x \in L$ zu entscheiden.

Um $L \in \mathrm{NP}^B \cap \mathrm{coNP}^B$ deterministisch zu entscheiden, haben wir also das Promise des komplementär akzeptierenden Maschinenpaars ausgenutzt, um ›gezielt‹ die eigentlich zu große Lücke $e(i)$ nach dem einzigen enthaltenden Wort zu durchsuchen. Auf dieser Idee bauen wir im nächsten Abschnitt auf.

4.2 Modifikation des Verfahrens

Wir werden nun das eben vorgestellte Verfahren modifizieren, um das eigentliche Ziel zu erreichen: Ein Orakel W konstruieren sodass UP ohne vollständige Sprache, und $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$.

In Rückblick auf das Verfahren bietet es sich sofort an, anstatt $\mathrm{P} \neq \mathrm{NP}$ über eine Diagonalisierung zu erreichen, stattdessen UP ohne vollständige Sprachen zu sichern. Schon im vorigen Kapitel haben wir gesehen, wie eine solche Diagonalisierung aussehen kann.

In Erinnerung an den stufenweisen Aufbau in den $e(\cdot)$-Lücken werden wir die UP-Zeugensprache D_j aus dem vorigen Kapitel deshalb nun auf den Stufenaufbau anpassen. Wir definieren

$$D'_j(W) =_{\mathrm{df}} \{1^n \mid s \geq 1,\, n = e(p_j^s), \exists z : |z| = n \wedge z \in W\}.$$

Auch das im vorigen Kapitel eingeführte Lemma 8 können wir für die kommende Konstruktion anpassen.

Lemma 16. *Sei $j, k \in \mathbb{N}$ beliebig, $i = p_j^s$, $n =_{\mathrm{df}} e(i)$ hinreichend groß sodass $2^n - 2 > 4\, t_j(t_k(n))$, und $C \subseteq \Sigma^*$ mit keinen Wörtern der Länge n in C.*

Es existiert eine Erweiterung $E \subseteq \Sigma^n$ sodass einer der folgenden Fälle eintritt:

(i) $|E| = 2$ *und* $N_j^{C \cup E}$ *akzeptiert* $T_k(1^n)$ *auf zwei Rechenwegen.*

(ii) $|E| \leq 1$ *mit* $1^n \in D'_j(C \cup E) \Leftrightarrow T_k(1^n) \notin L(N_j^{C \cup E})$.

Beweis. Folgt analog zur Beweisführung wie bei Lemma 8. □

Es bleibt zu zeigen, dass auch mit dieser Diagonalisierung $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ möglich bleibt. In dem skizzierten deterministischen Algorithmus des vorigen Abschnitts konnten wir beobachten, dass zunächst nicht möglich war, die gesamte Stufe $e(i)$ in deterministisch polynomieller Zeit nach Wörtern zu durchsuchen. Dafür aber konnten wir durch Ausnutzen des Promises eines NP∩coNP-Paares von Information über *kein* Wort in dieser Stufe auf Information über *ein* Wort der Stufe schließen. (Oder wie in Fall (a) darauf schließen, dass dieses Wort für die Berechnung irrelevant ist.)

Es liegt nahe, dass wir iterativ schrittweise von n bekannten Wörtern auf $n+1$ bekannte Wörter schließen können. Letzteres gilt nur falls das entsprechende Paar überhaupt

mit den n Wörtern komplementär akzeptiert. Existiert kein akzeptierender Rechenweg, können wir nicht über `AKZ` die relevanten Orakelfragen berechnen.

Wir starten also mit W', den Wörtern der Länge $e(0), \ldots, e(i-1)$ in W, und einer leeren Menge $E' = \emptyset$, die als ›Approximation‹ der Lücke $e(i)$ dient. Schrittweise bestimmen wir den akzeptierenden Weg auf Orakel $A \cup W' \cup E'$, und testen wie im Algorithmus des vorigen Kapitels, ob (a) alle relevanten Orakelwörter bestimmt wurden, oder ob (b) ein Wort w der Länge $e(i)$ erfragt wurde, was in W aber nicht in E' liegt. Dieses können wir zu E' hinzufügen, und haben ein neues Wort w gewonnen. Wir wiederholen die Schleife mit neuem E'; die Menge E' nähert sich dann jede Iteration an $W^{=e(i)}$ an.

Hieraus ergeben sich zwei Voraussetzungen: Einerseits muss in der Konstruktion $W^{=e(i)}$ nur endlich groß sein, damit der eben skizzierte Algorithmus in den Laufzeitschranken terminiert.

Andererseits muss unbedingt darauf geachtet werden, dass zu jedem E' im Algorithmus das entsprechende Paar auch immer komplementär akzeptiert, und so sich ein akzeptierender Rechenweg zur Untersuchung anbietet. Wir formalisieren diese wichtige Voraussetzung in folgender Aussage:

> *Sei N_a^W, N_b^W ein komplementär akzeptierendes Maschinenpaar. Für hinreichend lange Wörter x, sowie eindeutigem i mit $e(i-1) < \log|x| \leq e(i)$, gilt*
>
> $$N_a^{A \cup W' \cup E'}(x) \text{ akz.} \Leftrightarrow N_b^{A \cup W' \cup E'}(x) \text{ ablehnt} \qquad (*)$$
>
> *wobei $E' \subseteq W^{=e(i)}$ und $W' = W^{=e(0)} \cup W^{=e(1)} \cup \cdots \cup W^{=e(i-1)}$.*

Der Beweis des folgenden Satzes greift nun die bisher angesprochenen Ideen auf, und führt diese formal präzise aus.

Satz 17. *Es existiert ein Orakel W sodass*

(i) $\mathrm{P}^W = \mathrm{NP}^W \cap \mathrm{coNP}^W$*, und*

(ii) keine $\leq_{\mathrm{m}}^{\mathrm{p}}$-vollständige Sprache für UP^W existiert.

Beweis. Wie angekündigt erfolgt die Konstruktion von W durch stufenweises Füllen der Lücken $e(\cdot)$. In jeder Stufe i der Konstruktion definieren wir also die Lücke $e(i)$ in A. Sei nun W_i definiert als das Orakel nach Stufe i, in dem also bereits Wörter der Länge $e(0), \ldots e(i)$ festgesetzt sind. Insbesondere $W_i = A \cup W^{=e(0)} \cup W^{=e(1)} \cup \cdots \cup W^{=e(i)}$ (vgl. auch mit $A \cup W'$ in $(*)$).

In der Konstruktion versuchen wir, jeweils abzählbar unendlich viele Vorgaben zweier verschiedener Typen zu *erfüllen*:

- Erfüllen wir Vorgabe $c_{a,b}$ mit $a \neq b$, sichern wir, dass die Maschinen N_a^W und N_b^W nicht komplementär arbeiten.
- Analog sichern wir durch Erfüllen von $t_{j,k}$, dass N_j^W nicht kategorisch arbeitet, oder T_k keine Reduktion von $D'_j(W)$ auf $L(N_j^W)$ realisiert.

Sei $\mathcal{R} =_{\text{df}} \{c_{a,b} \mid a,b \in \mathbb{N}, a \neq b\} \cup \{t_{j,k} \mid j,k \in \mathbb{N}\} = \{c_{(0)}, c_{(1)}, \ldots, t_{(0)}, t_{(1)}, \ldots\}$ die Menge der abzählbar unendlich vielen Vorgaben. Wir ordnen die Vorgaben in $\mathcal{R}$ folgendermaßen durch die strikte Wohlordnung $\prec$ an:

$$c_{(0)} \prec t_{(0)} \prec c_{(1)} \prec t_{(1)} \prec c_{(2)} \prec t_{(2)} \prec \cdots$$

Damit ist $(\mathcal{R}, \prec)$ ordnungsisomorph zu $(\mathbb{N}, <)$; jede solche Ordnung $\prec$ ist für die Konstruktion günstig. Insbesondere ist dadurch gewährleistet, dass für beliebiges $r \in \mathcal{R}$ die Menge $\{r' \in \mathcal{R} \mid r' \prec r\}$ an ›priorisierten‹ Vorgaben endlich ist.

Noch nicht erfüllte Vorgaben $c_{a,b}$ nennen wir *in Stufe i gefährdet*, wenn ein x existiert sodass

$$\max\{t_a(|x|), t_b(|x|)\} < e(i+1), \quad N_a^{W_{i-1}}(x) \text{ akz.} \Leftrightarrow N_b^{W_{i-1}}(x) \text{ akz.} \qquad (**)$$

Solche gefährdete $c_{a,b}$ werden wir in der Konstruktion erfüllen, indem wir in Stufe i keine Wörter hinzufügen. Da die in höheren Stufen eingesetzten Wörter nicht von $N_a^{W_i}(x)$ bzw. $N_b^{W_i}(x)$ erfragt werden können, werden auch N_a^W und N_b^W selbes x nicht mehr komplementär akzeptieren.

Wir nennen nicht erfüllte $c_{a,b}$ *in Stufe i durch E' gefährdet*, wenn $E' \subseteq \Sigma^{e(i)}$, und $(**)$ gilt, dabei aber W_{i-1} durch $W_{i-1} \cup E'$ ausgetauscht wurde. Analog erfüllen wir durch E' gefährdete $c_{a,b}$ durch Hinzufügen aller Wörter in E'.

Nicht erfüllte Vorgaben $t_{j,k}$ nennen wir *in Stufe i gefährdet*, wenn

$$i = p_j^s, \quad 2^{e(i)} - 2 > 4\, t_j(t_k(e(i))), \quad t_j(t_k(e(i))) < e(i+1). \qquad (***)$$

Ist also $t_{j,k}$ gefährdet, dann ist einerseits $n =_{\text{df}} e(i)$ hinreichend groß wie Lemma 16 fordert, andererseits haben – analog zur Argumentation bei $c_{a,b}$ – die in höheren Stufen eingesetzten Wörter keinen Einfluss auf das Akzeptierverhalten von $N_j^{W_i}(T_k(1^n))$. Um $t_{j,k}$ zu erfüllen, bestimmen wir also entsprechende Erweiterung $E \subseteq \Sigma^n$, setzen *idealerweise* die Wörter von E in die Stufe i ein. Damit erzielen wir entweder (i) dass $N_j^W(1^n)$ nicht kategorisch akzeptiert, oder (ii) dass T_k keine Reduktion von $D'_j(W)$ auf $L(N_j^W)$ realisiert.

Wie schon angedeutet, liegt die Schwierigkeit der Konstruktion darin, jede Vorgabe $t_{j,k}$ zu erfüllen, gleichzeitig aber auch die Bedingung $(*)$ zu erhalten. Deshalb können wir beim Erfüllen von einem gefährdetem $t_{j,k}$ nicht ›*greedy*‹ alle Wörter in E der Stufe hinzufügen. Gerade wenn $E = \{w, v\}$ wäre ansonsten nur gesichert, dass komplementäre N_a^W, N_b^W zwar mit dem Orakel W_{i-1} komplementär akzeptieren, nicht aber mit $W_{i-1} \cup \{w\}$. Damit wäre dann nicht möglich, W_i aus W_{i-1} zu ermitteln, wie im Algorithmus vor dem Satz skizziert wurde.

Deshalb prüfen wir für alle $E' \subsetneq E$, ob durch E' ein $c_{a,b} \prec t_{j,k}$ gefährdet ist. Falls ein solches E' existiert, müssen wir die Erfüllung von $t_{j,k}$ abbrechen, und erfüllen stattdessen sofort die durch E' gefährdete Vorgaben. Erst wenn keine ›priorisierten‹ Vorgaben, also bezüglich $\prec$ kleinere Vorgabe gefährdet ist, können wir E vollständig einsetzen, $t_{j.k}$

erfüllen, und $(*)$ aufrecht halten. Damit bleiben gewissermaßen auch nur $c_{a,b}$ nicht erfüllt, für die (N_a, N_b) ›inhärent‹ komplementär akzeptierenden Maschinenpaare sind.

Die Konstruktion: Mit diesem Vorgehen ergibt sich folgende stufenweise Konstruktion. Wir sichern $W_0 \subseteq W_1 \subseteq W_2 \subseteq \ldots$, das fertige Orakel ist dann $W =_{\text{df}} \bigcup_{i=0}^{\infty} W_i$. Wir starten mit $W_0 =_{\text{df}} A$ und geben eine Vorschrift an, die W_i aus W_{i-1} definiert:

(1) Wir versuchen, die bezüglich $\prec$ kleinste, gefährdete Vorgabe $r \in \mathcal{R}$ zu erfüllen. Existiert kein gefährdetes r, setze $W_i =_{\text{df}} W_{i-1}$.

(2) Ist $r = c_{a,b}$, setze $W_i =_{\text{df}} W_{i-1}$ und markiere $c_{a,b}$ als erfüllt.

(3) Ist $r = t_{j,k}$, gilt $(***)$ und Voraussetzungen des Lemma 16 sind erfüllt. Bestimme $E \subseteq \Sigma^{e(i)}$ über das Lemma (mit W_{i-1} als Wahl für C). Wähle, falls möglich, ein $E' \subsetneq E$ aus, sodass ein $c_{a,b} \prec t_{j,k}$ durch E' gefährdet ist. Existieren mehrere solche E', wähle jene Teilmenge, durch die das gefährdete $c_{a,b}$ minimal ist.

   (a) Existiert ein solches E', setze $W_i =_{\text{df}} W_{i-1} \cup E'$ und markiere $c_{a,b}$ als erfüllt.

   (b) Andernfalls, wenn keine solche echte Teilmenge existiert, setze $W_i =_{\text{df}} W_{i-1} \cup E$. Ist E durch Fall (i) des Lemmas entstanden (also $N_j^{W_i}(1^n)$ nicht kategorisch), markiere alle $t_{j,0}, t_{j,1}, \ldots$ als erfüllt.

   Ansonsten, wenn Fall (ii) des Lemmas eingetreten ist (also T_k nicht die Reduktion zwischen $D'_j(W_i)$ und $L(N_k^{W_i})$ realisiert), markiere nur $t_{j,k}$ als erfüllt.

Zunächst argumentieren wir für Aussage (ii) des Satzes. Wir starten mit einer Beobachtung über die t-Vorgaben.

Behauptung 18. *Alle $t_{j,k} \in \mathcal{R}$ werden erfüllt.*

Beweis. Angenommen, es werden nicht alle $t_{j,k}$ erfüllt. Sei dann $t_{j,k}$ die bezüglich $\prec$ kleinste nicht erfüllte t-Vorgabe.

Zunächst zeigen wir, dass $t_{j,k}$ in unendlich vielen Stufen $i_1 < i_2 < \ldots$ gefährdet ist: Für unendlich viele i ist $i = p_j^s$, und ab hinreichend großem i sind auch die beiden Ungleichungen von $(***)$ erfüllt.

Nach Konstruktion wird dann also in Stufen $i_1, i_2, \ldots$ immer eine verletzte Vorgabe $r \preceq t_{j,k}$ als erfüllt markiert. Aus der Definition von $(\mathcal{R}, \prec)$ geht hervor, dass nur endlich viele Vorgaben kleiner als $t_{j,k}$ sind. Ab hinreichend großer Stufe i_n existieren dann also keine unerfüllten $r \prec t_{j,k}$ mehr, insbesondere dann keine kleineren verletzten Vorgaben, und $t_{j,k}$ wird widersprüchlicherweise in Fall (3a) der Stufe i_n erfüllt. □

Die folgende Behauptung zeigt nun, dass wir durch Erfüllen der t-Vorgaben tatsächlich die vor der Konstruktion angedeuteten gewünschten Eigenschaften in W sichern.

Behauptung 19. *Seien $j, k \in \mathbb{N}$ beliebig.*

(i) Für kategorische N_j^W realisiert T_k nicht die Reduktion von $D'_j(W)$ auf $L(N_j^W)$.

(ii) Für kategorische N_j^W ist $D'_j(W) \in \mathrm{UP}^W$.

(iii) Es existieren keine $\leq_{\mathrm{m}}^{\mathrm{p}}$-vollständigen Sprachen für UP^W.

Beweis. *Zu (i):* Sei i die Stufe in der $t_{j,k}$ erfüllt wurde; diese existiert nach voriger Behauptung. In dieser Stufe i ist dann also (3) der Konstruktion eingetreten.

Wir zeigen zunächst, dass E so gewählt wurde, dass Fall (ii) des Lemmas 16 eingetreten ist. Angenommen, E ist durch Fall (i) entstanden, dann würde (mit $W_i = W_{i-1} \cup E$) die Maschine $N_j^{W_i}(y)$ auf zwei verschiedenen Rechenwegen annehmen, wobei $y = T_k(1^{e(i)})$. Insbesondere war $t_{j,k}$ in Stufe i gefährdet, also gilt nach $(***)$

$$t_j(|y|) \leq t_j(t_k(e(i))) < e(i+1).$$

Es können also keine Fragen der Länge $\geq e(i+1)$ gestellt werden, damit werden bei der Berechnung von $N_j^W(y)$ die Orakelfragen genauso beantwortet wie von Orakel W_i. Also wird auch $N_j^W(y)$ auf zwei Rechenwegen annehmen; Widerspruch zur Wahl von N_j^W.

In Stufe i ist also E durch Fall (ii) des Lemmas entstanden, also

$$1^{e(i)} \in D'_j(W_i) \;\Leftrightarrow\; T_k(1^{e(i)}) \notin L(N_j^{W_i}).$$

Es ist $1^{e(i)} \in D'_j(W_i) \;\Leftrightarrow\; 1^{e(i)} \in D'_j(W)$: Einerseits ist $D'_j(W_i) \subseteq D'_j(W)$, andererseits werden keine weiteren Wörter der Länge $e(i)$ in höheren Stufen eingesetzt, damit $1^{e(i)} \notin D'_j(W_i) \;\Rightarrow\; 1^{e(i)} \notin D'_j(W)$

Es ist $T_k(1^{e(i)}) \in L(N_j^{W_i}) \;\Leftrightarrow\; T_k(1^{e(i)}) \in L(N_j^W)$: Nach gleicher Argumentation über der Laufzeit werden die Orakelfragen von W bzw. W_i gleich beantwortet, und die Maschinen akzeptieren die Eingabe äquivalent.

Damit gilt

$$1^{e(i)} \in D'_j(W) \;\Leftrightarrow\; 1^{e(i)} \in D'_j(W_i) \;\Leftrightarrow\; T_k(1^{e(i)}) \notin L(N_j^{W_i}) \;\Leftrightarrow\; T_k(1^{e(i)}) \notin L(N_j^W),$$

also realisiert T_k keine Reduktion von $D'_j(W)$ auf $L(N_j^W)$.

Zu (ii): Wir zeigen, dass $|W^{=e(i)}| \leq 1$ für alle $i = p_j^s$. Die Behauptung folgt dann analog wie im Beweis von Beobachtung 6(iii). Angenommen $|W^{=e(i)}| > 1$ für eine Potenz $i = p_j^s$. Da $W^{=e(i)} = W_i \setminus W_{i-1}$ untersuchen wir die Fälle, wie W_i definiert wurde.

Wird W_i in (1) bzw. (2) der Konstruktion definiert, dann $W_i = W_{i-1}$, $|W_i^{=e(i)}| = 0$; Widerspruch zur Annahme.

Wenn Fall (3) eintreten sollte, war nur ein Fall $t_{j,k}$ mit beliebigem $k \in \mathbb{N}$ gefährdet. (Für alle k, alle $j' \neq j$ ist Vorgabe $t_{j',k}$ nicht gefährdet, ist ja $p_{j'}^s \neq i$ für alle s.) Aus der Behauptung von Lemma 16 ist ersichtlich, dass in Fall (3) ein E mit $|E| \leq 2$ bestimmt wurde. Wird nun W_i in (3a) definiert, ist $W_i \setminus W_{i-1} = E' \subsetneq E$, also $|W_i^{=e(i)}| < |E| \leq 2$; Widerspruch zur Annahme.

Andernfalls, wenn W_i in (3b) definiert wird, wurde auch $t_{j,k}$ in Stufe i als erfüllt markiert. Aus dem Beweis von (i) folgt, dass E durch Lemma 16(ii) entstanden ist. Also auch $|E| \leq 1$; Widerspruch zur Annahme.

Zu (iii): Angenommen, es existiert ein vollständiges L für UP^W. Ähnlich wie im Beweis von Behauptung 14(i) existiert $j \in \mathbb{N}$ sodass N_j^W kategorisch, und $A \leq_{\mathrm{m}}^{\mathrm{p}} L(N_j^W)$ für alle $A \in \mathrm{UP}^W$. Da N_j^W kategorisch, ist $D'_j \in \mathrm{UP}^W$ nach Aussage (ii), nach Annahme dann $D'_j \leq_{\mathrm{m}}^{\mathrm{p}} L(N_j^W)$.

Widerspruch zu (i), denn kein berechenbarer Transduktor kann die Reduktion realisieren. Also existiert keine für UP^W vollständige Sprache. □

Für Aussage (i) des Satzes starten wir wieder mit einer analogen Behauptung über die c-Vorgaben. Wie angekündigt können wir dann ableiten, dass wir durch die Konstruktion die Bedingung $(*)$ erfüllt haben.

Behauptung 20. *Ist die Vorgabe $c_{a,b}$ in einem beliebigen Schritt erfüllt worden, akzeptiert das Paar N_a^W und N_b^W nicht komplementär.*

Beweis. Sei die Vorgabe $c_{a,b}$ in Schritt i erfüllt. Ohne Beschränkung können wir annehmen, dass $c_{a,b}$ durch E gefährdet war (andernfalls wählen wir $E = \emptyset$ und erreichen $(**)$). Demnach existiert x welches von N_a und N_b mit Orakel $W_{i-1} \cup E$ nicht komplementär akzeptiert wird.

Aus der Konstruktion ist ersichtlich, dass $c_{a,b}$ nur dann in Stufe i als erfüllt markiert wurde, wenn $W_i = W_{i-1} \cup E$ gesetzt wurde. Also akzeptieren auch $N_a^{W_i}(x)$ und $N_b^{W_i}(x)$ nicht komplementär.

Da $c_{a,b}$ gefährdet war, gilt insbesondere auch $\max\{t_a(|x|), t_b(|x|)\} < e(i+1)$. Wie im vorigen Beweis werden bei der Berechnung von $N_a^{W_i}(x)$ also höchstens Wörter der Länge $t_a(|x|) < e(i+1)$ gestellt. Das Orakel W beantwortet diese Fragen genauso wie W_i, also akzeptiert $N_a^W(x)$ äquivalent zu $N_a^{W_i}(x)$. Analog dann für N_b, und das Paar $N_a^W(x)$ und $N_b^W(x)$ akzeptiert nicht komplementär. □

Die Bedingung $(*)$ konkretisieren wir in folgender Behauptung.

Behauptung 21. *Sei N_a^W, N_b^W ein komplementär akzeptierendes Maschinenpaar. Ab hinreichend großem i ist sichergestellt, dass für alle x mit $e(i-1) < \log|x| \leq e(i)$ auch*

(i) $N_a^W(x)$ akzeptiert genau dann wenn $N_a^{W_i}(x)$ akzeptiert, analog für N_b.

(ii) für beliebige $E' \subseteq W_i \setminus W_{i-1}$ die Berechnung $N_a^{W_{i-1} \cup E'}(x)$ genau dann akzeptiert wenn $N_b^{W_{i-1} \cup E'}(x)$ ablehnt.

Beweis. Sei N_a, N_b ein relativ zu W komplementär akzeptierendes Maschinenpaar. Sofort können wir aus Kontraposition von voriger Behauptung festhalten, dass $c_{a,b}$ in keiner Stufe erfüllt wird.

Zunächst zeigen wir, dass i_0 existiert sodass für alle $i > i_0$ die beiden folgenden Bedingungen gelten.

(a) $\max\{t_a(2^{e(i)}), t_b(2^{e(i)})\} < e(i+1)$.

(b) Alle kleineren Vorgaben $r \prec c_{a,b}$, die in irgend einer Stufe der Konstruktion erfüllt werden, werden vor Stufe i erfüllt.

Sei hierfür $c =_{\text{df}} \max\{a, b\}$, $n =_{\text{df}} e(i)$, dann ist

$$\max\{t_a(2^{e(i)}), t_b(2^{e(i)})\} = 2^{cn} + c \in o(2^{2^n}) = o(e(i+1)),$$

Bedingung (a) also ab hinreichend großem i_0 erfüllt.

Die Menge an Vorgaben $\{r' \in \mathcal{R} \mid r' \prec c_{a,b}\}$ ist aufgrund Definition von $(\mathcal{R}, \prec)$ endlich. Insbesondere existieren dann also auch nur endlich viele kleinere Vorgaben $r \prec c_{a,b}$, die jemals erfüllt werden. Also lässt sich auch ein i_0 angeben, zu denen alle endlich vielen r erfüllt sind, sollten sie jemals erfüllt werden.

Zu (i): Wir beobachten, dass aus der Voraussetzung die Eingabelänge $|x|$ auf $\leq 2^{e(i)}$ beschränkt ist. Wie im vorigen Beweis werden bei der Berechnung von $N_a^W(x)$ also höchstens höchstens Wörter der Länge $t_a(|x|) \leq t_a(2^{e(i)}) < e(i+1)$ gestellt. Das Orakel W_i beantwortet diese Fragen genauso wie W, also akzeptiert $N_a^{Z_i}(x)$ äquivalent. Analog dann für N_b.

Zu (ii): Angenommen, es existieren geeignete $i > i_0$, $E' \subseteq W_i \setminus W_{i-1}$, x mit $e(i-1) \leq \log|x| < e(i)$ sodass (ii) nicht gilt, also $N_a^{W_{i-1} \cup E'}(x)$ und $N_a^{W_{i-1} \cup E'}(x)$ nicht komplementär akzeptieren. Wieder ist $|x| \leq 2^{e(i)}$.

Ist $E' = \emptyset$, dann wäre aufgrund (a)

$$\max\{t_a(|x|), t_b(|x|)\} \leq \max\{t_a(2^{e(i)}), t_b(2^{e(i)})\} < e(i+1),$$

also die Vorgabe $c_{a,b}$ in Stufe i gefährdet, aufgrund (b) die kleinste gefährdete Vorgabe. Fall (1) wählt $r = c_{a,b}$, in Fall (2) wird dann $c_{a,b}$ widersprüchlicherweise erfüllt.

Sei also $E' \neq \emptyset$. Ist $E' = W_i \setminus W_{i-1}$, dann auch $W_{i-1} \cup E' = W_i$. Nach Annahme akzeptieren also $N_a^{W_i}$ und $N_b^{W_i}$ nicht komplementär, nach voriger Aussage (i) dann N_a^W und N_b^W nicht komplementär. Widerspruch zur Wahl von N_a, N_b.

Sei also $E' \subsetneq W_i \setminus W_{i-1}$. Insbesondere ist dann $W_i \neq W_{i-1}$, also wurde in Stufe i Fall (3) der Konstruktion erreicht. Sei hierfür $r = t_{j,k}$ die in Stufe i gewählte Vorgabe. Wieder aufgrund (b) $c_{a,b} \prec t_{j,k}$. (Der Fall $c_{a,b} \succ t_{j,k}$ ist ausgeschlossen, weil ja $t_{j,k}$ nach Behauptung 18 in der Konstruktion immer erfüllt wird.)

Wir zeigen, dass in Fall (3) die Teilmenge E' gewählt wird, und so widersprüchlicherweise $c_{a,b}$ erfüllt wird: Einerseits ist nach (a) und Wahl von E' und x die Vorgabe $c_{a,b} \prec t_{j,k}$ durch E' gefährdet. Andererseits ist für alle anderen Teilmengen $E'' \subsetneq E$ mit $E'' \neq E'$ keine kleinere Vorgabe gefährdet. Alle $c_{a',b'} \prec c_{a,b}$ sind nach Bedingung (b) entweder schon erfüllt, oder werden nie erfüllt (insbesondere nicht aufgrund Gefährdung durch E'' in Stufe i).

Damit tritt (3a) mit E' und $c_{a,b}$ ein, $c_{a,b}$ wird widersprüchlicherweise erfüllt. □

Mit voriger Behauptung können wir nun die Behauptung (i) des Satzes zeigen. Wir orientieren uns dabei an das vor dem Satz skizzierte Verfahren, um mit zwei komplementären Maschinen von W_{i-1} auf W_i zu schließen.

Behauptung 22. *Es ist* $\mathrm{NP}^W \cap \mathrm{coNP}^W \subseteq \mathrm{P}^W$.

Beweis. Sei $L \in \mathrm{NP}^W \cap \mathrm{coNP}^W$, dann existieren komplementär akzeptierende N_a^W, N_b^W mit $L(N_a^W) = L$. Insbesondere existiert i_0, sodass für alle $i > i_0$ die in Behauptung 21 angegebenen Aussagen (i) und (ii) gelten.

Im Folgenden betrachten wir deshalb nur noch Eingaben x mit $e(i_0) < \log|x|$. Alle endlich vielen kürzeren Eingaben können wir in linearer Zeit entscheiden. In der folgenden Berechnungsvorschrift entscheiden wir folgendermaßen für entsprechend große x, ob $x \in L$:

(1) Bestimme kleinstes $i > i_0$ sodass $\log|x| \leq e(i)$.

(2) Es ist $e(i-1) < \log|x|$, wir können also die Menge

$$W' = W^{=e(0)} \cup W^{=e(1)} \cup \cdots \cup W^{=e(i-1)} \quad (\text{insb. } W' \cap A = \emptyset)$$

über maximal

$$2^{e(0)} + 2^{e(1)} + \cdots + 2^{e(i-1)} \leq i \cdot 2^{\log|x|} \leq i|x| \in \mathcal{O}(|x|^2)$$

Orakelfragen der Länge $e(0)$, $e(1)$, …, $e(i-1)$ berechnen. Insbesondere $|z| \leq |x|$ für alle $z \in W'$, und $W' \cup A = W_{i-1}$, wie es in der Konstruktion definiert wurde.

(3) Da $i > i_0$ können wir aus Behauptung 21(i) folgern, dass $N_a^W(x)$ akzeptiert genau dann wenn $N_a^{W_i}(x)$ akzeptiert.

Es reicht also aus, eine entscheidungsäquivalente Erweiterung $E' \subseteq W_i \setminus W_{i-1}$ zu bestimmen, sodass sich N_a bzw. N_b mit W_i nicht anders verhalten als mit $W_{i-1} \cup E'$. Orakelfragen der Länge $\neq e(i)$ werden von W_{i-1} ohnehin schon genauso wie W_i beantwortet. Es ist also schon hinreichend, ein E' anzugeben, dass Fragen der Länge $e(i)$ genauso wie W_i (bzw. genauso wie W) beantwortet.

Wir starten mit $E' \leftarrow \emptyset$, und sichern in jedem Iterationsschritt $E' \subseteq W^{=e(i)} = W_i \setminus W_{i-1}$.

(4) Da $E' \subseteq W_i \setminus W_{i-1}$ wird nach Behauptung 21(ii) genau eine der beiden Maschinen mit Orakel $W_{i-1} \cup E'$ halten.

Später geben wir eine Konstruktion von $\mathtt{AKZ}' \in \mathrm{FP}^W$ an, für die

$$\mathtt{AKZ}'(c, 1^{t_c(|x|)}, x, W') =_{\mathrm{df}} \text{Rechenweg von } N_c^{A \cup W'}(x)$$

falls die Berechnung akzeptiert, ansonsten $\mathtt{AKZ}'(\cdot) =_{\mathrm{df}} \varepsilon$. Setze

$$N \leftarrow N_a, \quad Q \leftarrow \text{Die Orakelfragen von } \mathtt{AKZ}'(a, 1^{t_a(|x|)}, x, W' \cup E')$$

wenn $\texttt{AKZ}'(a, 1^{t_j(|x|)}, x, W' \cup E') \neq \varepsilon$, sonst

$$N \leftarrow N_b, \quad Q \leftarrow \text{die Orakelfragen von } \texttt{AKZ}(b, 1^{t_b(|x|)}, x, W' \cup E').$$

Also ist $N \in \{N_a, N_b\}$ die akzeptierende Maschine, $Q \in \Sigma^*$ die Menge an Orakelfragen, die auf dem akzeptierenden Rechenweg von N mit Orakel $W_{i-1} \cup E'$ gestellt wurden.

(a) Ist $E' = Q \cap W^{=e(i)}$, wird sich, wie bereits argumentiert, dann $N(x)$ mit Orakel $W_{i-1} \cup E'$ nicht anders verhalten, als mit Orakel W.

Wenn $N = N_a$, wird also $N_a^W(x)$ akzeptieren, dann $x \in L$, und terminiere akzeptierend. Andernfalls, wenn $N = N_b$, wird $N_b^W(x)$ akzeptieren, also $x \in \overline{L}$, und terminiere ablehnend.

(b) Ist $E' \subsetneq Q \cap W^{=e(i)}$, setze $E' \leftarrow Q \cap W^{=e(i)}$ und springe zu (4).

Der angegebene Algorithmus hält die polynomielle Laufzeit ein: In jeder Iteration wird E' echt größer, und weiterhin ist $|E'| \leq |W^{=e(i)}| \leq 2$, also terminiert der Algorithmus nach höchstens zwei Iterationen. Weiterhin ist die Eingabegröße von $\texttt{AKZ}'$ in Schritt (4) polynomiell beschränkt: Die $\mathcal{O}(|x|^2)$ vielen Wörter in W' haben Länge $\mathcal{O}(|x|)$, die höchstens konstant vielen Wörter in E' haben Länge $\leq \max\{t_a(|x|), t_b(|x|)\}$. Damit ist $L \in \mathrm{P}^W$.

Es verbleibt zu zeigen, dass $\texttt{AKZ}' \in \mathrm{FP}^W$. Sei hierfür N_u^W die nichtdeterministische Orakel-Maschine mit folgender Arbeitsweise auf Eingabe $\langle c, 1^{t_c(|x|)}, x, W' \rangle$:

> Führe die Berechnung von $\texttt{AKZ}(c, 1^{t_c(|x|)}, x)$ aus. Beantworte aber Orakelfragen während der Berechnung mit Länge $e(\cdot)$ durch W', alle anderen Orakelfragen werden an W weitergeleitet.

Da die Laufzeit von $\texttt{AKZ}$ polynomiell beschränkt war, hält N_u^W in polynomieller Laufzeitschranke. Insbesondere ist das Durchsuchen von W' bei Orakelfragen der Länge $e(\cdot)$ in polynomieller Zeit abhängig von der Eingabelänge möglich.

Die nichtdeterministische Polynomialzeit-Maschine N_u^W berechnet die Funktion $\texttt{AKZ}'$: Da W ohne Wörter der Länge $e(\cdot)$ dem Orakel A entspricht, wird $N_u^W(c, 1^{t_c(|x|)}, x, W')$ die Berechnung $\texttt{AKZ}(c, 1^{t_c(|x|)}, x)$ mit Orakel $A \cup W'$ simulieren. Also wird der Rechenweg von $N_c^{A \cup W'}$ berechnet. Das entspricht $\texttt{AKZ}'(c, 1^{t_c(|x|)}, x, W')$ nach Definition.

Insbesondere stellt N_u^W keine Fragen der Länge $e(\cdot)$, damit verhält sich die Berechnung genauso wie relativ zum Orakel A. Da $\mathrm{NP}^A = \mathrm{P}^A$, existiert eine entsprechende äquivalente Maschine $P_{u'}^A$. Ohne Beschränkung stellt auch diese Maschine relativ zu A keine Fragen der Länge $e(\cdot)$, und $P_{u'}^A$ arbeitet identisch relativ zu W. Damit kann dann $\texttt{AKZ}'$ relativ zu W in deterministischer Polynomialzeit durch $P_{u'}^W$ berechnet werden. □

Das schließt den Beweis vom Satz ab: Die zu zeigenden Eigenschaften (i) bzw. (ii) von W folgen aus Behauptung 22 bzw. Behauptung 19(iii). □

5 Möglichkeiten und Grenzen des Verfahrens

Schon die Herleitung der Konstruktion von W aus der Technik von Baker et al. suggeriert, wie flexibel das Verfahren angepasst werden kann. So wurde im letzten Kapitel die für P $\neq$ NP verantwortliche Diagonalisierung ausgetauscht, um über eine geeignete Diagonalisierung die Hypothese UP zu erreichen. Es liegt nahe, dass dann auch andere von Pudlák definierten Hypothesen im Rahmen von Promise-Klassen erreicht werden können.

Insbesondere wird im nächsten Abschnitt zunächst demonstriert, wie das Verfahren mit einfachen Änderungen so erweitert werden kann, dass die Hypothesen ¬NP∩coNP, UP, und gleichzeitig Hypothese DisjNP gilt. Hierbei ist DisjNP die von Pudlák definierte Hypothese, dass kein vollständiges disjunktes Paar an NP-Sprachen existiert.

Der Rest des Kapitels versucht, allgemeine Voraussetzungen für die Durchführbarkeit des Verfahrens anzugeben. Einerseits zeigen die daraus resultierenden Grenzen der Konstruktion, dass die leicht stärkeren Promise-Klassen U_pP von NP getrennt werden können. Andererseits wird abschließend argumentiert, dass dieses Verfahren vermutlich nicht (trivial) in der Lage sein wird, mit noch stärkeren Promise-Klassen FewP oder DisjcoNP umzugehen.

5.1 Ergänzung: Auch DisjNP ohne vollständige Paare

Sind zwei Sprachen A, B in NP, sowie $A \cap B = \emptyset$, ist (A, B) ein disjunktes NP-Paar. Die Menge an disjunkten NP-Paaren bildet die Promise-Klasse DisjNP. Auf Maschinen bezogen ist mit dem Promise das Versprechen gegeben, dass zu $(A, B) \in$ DisjNP zwei nichtdeterministische Maschinen existieren, die *disjunkt* akzeptieren:

Definition 23. Die Komplexitätsklasse DisjNP entspricht den Paaren an Sprachen, die von disjunkt akzeptierenden nichtdeterministischen Polynomialzeit-Maschinen erkannt werden kann:

$$\mathrm{DisjNP} =_{\mathrm{df}} \{(L(N_a), L(N_b)) \mid a, b \in \mathbb{N},\ \forall x : N_a(x) \text{ lehnt ab} \vee N_b(x) \text{ lenht ab}\}.$$

Zwischen zwei Paaren lässt sich eine *Many-One-Poynomialzeit-Promise-Problem-Reduktion* $\leq^{\mathrm{pp}}_{\mathrm{m}}$ definieren:

$$(A, B) \leq^{\mathrm{pp}}_{\mathrm{m}} (C, D) \ \Leftrightarrow\ {}_{\mathrm{df}}\text{es existiert } f \in \mathrm{FP} \text{ sodass } f(A) \subseteq C,\ f(B) \subseteq D.$$

Entsprechend werden $\leq^{\mathrm{pp}}_{\mathrm{m}}$-vollständige disjunkte NP-Paare definiert. [Raz94]

Diese Definition der Reduktion zwischen zwei Paaren baut auf der Interpretation von NP-Paaren als *Promise-Problem* (hier abzugrenzen von Promise-*Klassen*) auf. Für $(\Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}}) \in$ DisjNP lässt sich beispielsweise fragen, ob ein Algorithmus effizient die Ja-Instanzen Π_{Y} von den Nein-Instanzen Π_{N} trennen kann. Insbesondere wird dabei aber

dem Algorithmus ›versprochen‹, dass die Eingabe im ›Definitionsbereich‹ $\Pi_Y \cup \Pi_N$ liegt. Für ›unerlaubte‹ Eingaben $x \notin \Pi_Y \cup \Pi_N$ kann sich der Algorithmus beliebig verhalten. Die Idee des Promise-Problems überträgt sich auf den gewählten Reduktionsbegriff: Von der Funktion f wird nur eine ›korrekte‹ Zuordnung für Eingaben $\Pi_Y \cup \Pi_N$ gefordert, für Eingaben $\overline{\Pi_Y \cup \Pi_N}$ kann sich f beliebig verhalten. Even, Selman und Yacobi motivierten insbesondere disjunkte NP-Paare in ihren Untersuchungen zu Public-Key-Kryptosystemen. [vgl. Gol06]

Razborov zeigte, dass aus der Existenz von optimalen Beweissystemen die Existenz von $\leq_m^{pp}$-vollständigen disjunkten NP-Paaren folgt. Pudláks definiert die Hypothese DisjNP dann analog zu UP als Hypothese, dass keine $\leq_m^{pp}$-vollständigen disjunkten NP-Paare existieren. Ähnlich wie $\mathsf{UP} \Rightarrow \mathsf{CON}$ ist auch $\mathsf{DisjNP} \Rightarrow \mathsf{CON}$. [vgl. Pud17, Sec. 5.1]

Nichtsdestotrotz zeigen Dose und Glaßer, dass die relativierten Hypothesen UP und DisjNP nicht gleich sind. Insbesondere existiert ein Orakel, relativ zu dem $\mathsf{DisjNP} \wedge \neg\mathsf{UP}$ gilt. [DG19, Cor. 4.2]

Dennoch ist wegen der Ähnlichkeit der beiden Hypothesen in ihrer Beziehung zu CON zu vermuten, dass auch ein Orakel für $\mathsf{DisjNP} \not\Rightarrow \mathsf{NP} \cap \mathsf{coNP}$ existiert. Tatsächlich lässt sich mit wenigen Erweiterungen jene Konstruktion verändern, dass wie angekündigt auch relativ zu einem Orakel $\mathsf{DisjNP} \wedge \mathsf{UP} \not\Rightarrow \mathsf{NP} \cap \mathsf{coNP}$. Hierfür geben wir zunächst eine Möglichkeit an, über Diagonalisierung ein vollständiges Paar auszuschließen. Anschließend wird eine Erweiterung der Konstruktionsvorschrift angegeben, und die behaupteten Eigenschaften des konstruierten Orakels nachgewiesen.

Wieder können wir uns bei den Reduktionsfunktionen auf Transduktoren ohne Zugriff auf das Orakel beschränken. Der Beweis erfolgt analog zu Lemma 5.

Lemma 24. *Es existieren genau dann $\leq_m^{pp}$-vollständige Paare in* DisjNP^O*, wenn auch $\leq_m^{pp,O}$-vollständige Paare in* DisjNP^O *existieren.*

Beweis. Wir zeigen wieder nur die Implikation von rechts nach links. Sei (A, B) $\leq_m^{pp,O}$-vollständig für DisjNP^O. Wir definieren

$$A' =_{df} \{\langle i, 1^{t_i(|x|)}, x\rangle \mid T_i^O(x) \in A\}, \quad \text{analog } B'.$$

Offenbar ist $(A', B') \in \mathrm{DisjNP}^O$. Weiterhin ist (A', B') auch $\leq_m^{pp}$-vollständig: Sei $(C, D) \in \mathrm{DisjNP}^O$, und sei f die FP^O-Funktion, welche die $\leq_m^{pp,O}$-Reduktion von (C, D) auf (A, B) realisiert. Insbesondere ist $f(C) \subseteq A$, $f(D) \subseteq B$. Sei T_i^O der Transduktor, der f simuliert.

Dann ist durch $f'(x) =_{df} \langle i, 1^{t_i(|x|)}, x\rangle$ eine FP-Funktion ohne Orakelzugriff gegeben, die (C, D) auf (A', B') reduziert:

$$x \in C \;\Rightarrow\; f(x) \in A \;\Rightarrow\; T_i^O(x) \in A \;\Rightarrow\; \langle i, 1^{t_i(|x|)}, x\rangle \in A' \;\Rightarrow\; f'(x) \in A',$$

und analog für $x \in D$, also $f'(C) \subseteq A'$, $f'(D) \subseteq B'$. □

Wir definieren folgendes Zeugensprachen-Paar:

$$X'_{a,b}(W) =_{\text{df}} \{1^n \mid s \geq 1,\, n = e(p^s_{a,b}), \exists z : |z| = n, z \in W, z \text{ gerade}\},$$
$$Y'_{a,b}(W) =_{\text{df}} \{1^n \mid s \geq 1,\, n = e(p^s_{a,b}), \exists z : |z| = n, z \in W, z \text{ ungerade}\}.$$

Wieder analog zu Beobachtung 6 halten wir fest, dass $(X'_{a,b}(W), Y'_{a,b}(W)) \notin \text{DisjNP}^W$ nur dann, wenn für eine Potenz $n = p^s_{a,b}$ sowohl ein gerades, als auch ein ungerades Wort der Länge n in W liegt.

Analog zu Lemma 16 geben wir nun an, wie ein Diagonalisierungsschritt für die DisjNP-Paare erfolgen kann.

Lemma 25. *Seien $a, b, k \in \mathbb{N}$ beliebig, q ein Polynom, was die Laufzeit von N_a und N_b beschränkt, $i = p^s_{a,b}$, $n =_{\text{df}} e(i)$ hinreichend groß sodass $2^n - 2 > 4\,q(t_k(n))$, und $C \subseteq \Sigma^*$ mit keinen Wörtern der Länge n in C.*

Es existiert eine Erweiterung $E \subseteq \Sigma^n$ sodass einer der folgenden Fälle eintritt:

(i) $|E| = 2$ *und sowohl* $N_a^{C \cup E}$ *also auch* $N_b^{C \cup E}$ *akzeptieren Eingabe* $T_k(1^n)$.

(ii) $E = \{u\}$, *u gerade, und* $1^n \in X'_{a,b}(C \cup E) \Leftrightarrow T_k(1^n) \notin L(N_a^{C \cup E})$.

(iii) $E = \{v\}$, *v ungerade, und* $1^n \in Y'_{a,b}(C \cup E) \Leftrightarrow T_k(1^n) \notin L(N_b^{C \cup E})$.

Beweis. Sei im Folgenden $y =_{\text{df}} T_k(1^n)$. Angenommen, zu gegebenen N_a, N_b, T_k, C existiert keine Erweiterung E, sodass Fall (i), (ii) oder Fall (iii) eintritt.

Da (ii) und (iii) nicht eintreten, wird für alle geraden $u \in \Sigma^n$ die Berechnung $N_a^{C \cup \{u\}}(y)$ akzeptieren, für alle ungeraden $v \in \Sigma^n$ die Berechnung $N_b^{C \cup \{v\}}(y)$ akzeptieren.

Identisch zur Beweisführung in Lemma 9 existiert dann eine Erweiterung $E = \{u, v\}$, sodass $N_a^{C \cup E}(y)$ annimmt, und $N_b^{C \cup E}(y)$ annimmt. Damit ist Fall (i) eingetreten, Widerspruch zur Annahme. □

Wir beobachten die Ähnlichkeit zwischen dem gerade bewiesenen Lemma, und Lemma 16, welches in der vorigen Konstruktion die Diagonalisierung für UP ermöglichte:

Tritt für N_a, N_b, T_k Fall (ii) oder (iii) auf, ist die Reduktion $(L(N_a), L(N_b)) \leq^{pp}_m (X', Y')$ über Transduktor T_k ausgeschlossen. (Lemma 16(ii) schloss aus, dass T_k die Reduktion von D'_j auf $L(N_j)$ realisierte.)

Tritt Fall (i) auf, akzeptieren N_a und N_b nicht mehr disjunkt, der Promise wurde zerstört, $(L(N_a), L(N_b)) \notin \text{DisjNP}$. (Analog hat Lemma 16(i) den UP-Promise von N_j zerstört.)

Insbesondere ist auch immer E so gewählt dass $|E| \leq 2$. Aus dieser Ähnlichkeit lässt sich nur durch wenige Veränderungen der Konstruktion von W in Satz 17 erreichen, dass statt UP die Klasse DisjNP keine vollständigen Sprachen besitzt.

Wie angekündigt können wir sogar die beiden Diagonalisierungen parallel durchführen, um die drei Eigenschaften gleichzeitig sicherzustellen.

Satz 26. *Es existiert ein Orakel W, relativ zu diesem*

(i) $\mathrm{P}^W = \mathrm{NP}^W \cap \mathrm{coNP}^W$, *und*

(ii) keine $\leq^{\mathrm{p}}_{\mathrm{m}}$*-vollständige Sprache für* UP^W *existiert, und*

(iii) kein $\leq^{pp}_{m}$*-vollständiges Paar für* DisjNP^W *existiert.*

Beweis. Es wird nur angegeben, wie die bisherige Konstruktion aus Satz 17 ergänzt werden muss, um die drei Eigenschaften zu erreichen.

Zunächst erweitern wir die Menge an Vorgaben, damit die DisjNP-Diagonalisierung durchgeführt wird. Sei

$$\begin{aligned}\mathcal{R} &=_{\mathrm{df}} \{c_{a,b} \mid a,b \in \mathbb{N}, a \neq b\} \cup \{t_{j,k} \mid j,k \in \mathbb{N}\} \cup \{d_{a,b,k} \mid a,b,j \in \mathbb{N}, a \neq b\} \\ &= \{c_{(0)}, c_{(1)}, \ldots, t_{(0)}, t_{(1)}, \ldots, d_{(0)}, d_{(1)}, \ldots\}\end{aligned}$$

die neue Menge an abzählbar unendlich vielen Vorgaben. Im Vergleich zur ursprünglichen Konstruktion wurden die Vorgaben $d_{a,b,k}$ hinzugefügt. Die Vorgabe $d_{a,b,k}$ soll dann absichern, dass das Paar $(L(N_a^W), L(N_b^W))$ nicht disjunkt ist, oder T_k keine Reduktion von $(X'_{a,b}(W), Y'_{a,b}(W))$ auf $(L(N_a^W), L(N_b^W))$ realisiert.

Wieder ordnen wir $\mathcal{R}$ isomorph zu $(\mathbb{N}, <)$ an:

$$c_{(0)} \prec t_{(0)} \prec d_{(0)} \prec c_{(1)} \prec t_{(1)} \prec d_{(1)} \prec c_{(2)} \prec \cdots$$

Analog zu $t_{j,k}$ nennen wir Vorgabe $d_{a,b,k}$ in Stufe i gefährdet, wenn

$$i = p^s_{a,b}, \quad 2^n - 2 > 4\,q(t_k(n)), \quad q(t_k(e(i))) < e(i+1),$$

wobei $q(n) =_{\mathrm{df}} \max\{t_a(n), t_b(n)\}$. Solche gefährdeten $d_{a,b,k}$ werden wir analog zu $t_{j,k}$ sicherstellen: Es ist $n =_{\mathrm{df}} e(i)$ hinreichend groß wie Lemma 25 fordert, wir bestimmen $E \subseteq \Sigma^n$ und versuchen wieder, die gesamte Erweiterung E in die Lücke $e(i)$ einzusetzen. Wieder analog zur Erfüllung von $t_{j,k}$ achten wir dabei darauf, $(*)$ nicht zu verletzen.

Wir erweitern die Vorschrift der Konstruktion von Satz 17 um einen weiteren Fall, orientieren uns dabei an Fall (3).

(4) Ist $r = d_{a,b,k}$ die kleinste gefährdete Vorgabe, bestimme $E \subseteq \Sigma^{e(i)}$ aus Lemma 25 (mit W_{i-1} als Wahl für C). Wähle unter allen Teilmengen $E' \subsetneq E$ jene aus, für die das durch E' gefährdete $c_{a,b} \prec d_{a,b,k}$ minimal ist.

   (a) Existiert ein solches E', setze $W_i =_{\mathrm{df}} W_{i-1} \cup E'$ und markiere $c_{a,b}$ als erfüllt.

   (b) Andernfalls, wenn keine solche echte Teilmenge existiert, setze $W_i =_{\mathrm{df}} W_{i-1} \cup E$. Ist E durch Fall (i) des Lemmas entstanden (also $L(N_a^W) \cap L(N_b^W) \neq \emptyset$), markiere alle $d_{a,b,0}, d_{a,b,1}, \ldots$ als erfüllt.

   Ansonsten, wenn Fall (ii) oder (iii) des Lemmas eingetreten ist (also T_k nicht die Reduktion von $(X'_{a,b}(W), Y'_{a,b}(W))$ auf $(L(N_a^W), L(N_b^W))$ realisiert), markiere nur $d_{a,b,k}$ als erfüllt.

Offenbar bleiben damit Behauptungen im Beweis von Satz 17 bestehen, höchstens Behauptung 21(ii) erfordert weitere Argumentation:

Insbesondere folgt aus $W_i \neq W_{i-1}$, dass Fall (3) *oder* Fall (4) der Konstruktion erreicht wurde. Für Fall (3) mit $r = t_{j,k}$ wurde im ursprünglichen Beweis schon gezeigt, dass dadurch Behauptung 21(ii) nicht verletzt wurde. Für Fall (4) mit $r = d_{a,b,k}$ erfolgt dann aber auch die Argumentation analog wie zu Fall (3) des Beweises von Behauptung 21(ii).

Weiterhin wird der im Beweis von Behauptung 22 angegebene Algorithmus immer noch in Polynomialzeit laufen: Wieder sind die Anzahl an Iterationen auf $|E| \leq |W^{=e(i)}| \leq 2$ beschränkt, egal ob E aus Lemma 16 (für UP) oder Lemma 25 (für DisjNP) entstanden ist.

Damit sind die Eigenschaften (i) und (ii) des Orakels W schon gezeigt. Es verbleibt, (iii) zu beweisen.

Behauptung 27. *Alle $d_{a,b,k} \in \mathcal{R}$ werden erfüllt.*

Beweis. Die Beweisführung erfolgt analog zu der von Behauptung 18. Insbesondere ist immer noch $\{r' \in \mathcal{R} \mid r' \prec d_{a,b,k}\}$ endlich. □

Behauptung 28. *Seien $a, b, k \in \mathbb{N}$ beliebig.*

(i) Für disjunkte $L(N_a^W)$, $L(N_b^W)$ realisiert T_k nicht die Reduktion von $(X'_{a,b}(W), Y'_{a,b}(W))$ auf $(L(N_a^W), L(N_b^W))$.

(ii) Für disjunkte $L(N_a^W)$, $L(N_b^W)$ ist $(X'_{a,b}(W), Y'_{a,b}(W)) \in \mathrm{DisjNP}^W$.

(iii) Es existieren keine $\leq_m^{pp}$-vollständigen Paare für DisjNP^W.

Beweis. Die Beweise der Teilaussagen erfolgen ganz analog zu denen von Behauptung 19.

Zu (i): Sei $L(N_a^W) \cap L(N_b^W) \neq \emptyset$, T_k beliebig, i die Stufe in der $d_{a,b,k}$ erfüllt wurde; diese existiert nach voriger Behauptung. In dieser Stufe i ist dann also (4) der Konstruktion eingetreten.

Angenommen, E ist durch Fall (i) des Lemmas 25 entstanden, dann würden (mit $W_i = W_{i-1} \cup E$) sowohl $N_a^{W_i}(y)$ als auch $N_b^{W_i}(y)$ akzeptieren, wobei $y = T_k(1^{e(i)})$. Insbesondere war $d_{a,b,k}$ in Stufe i gefährdet, über Argumentation über die Laufzeit werden dann auch $N_a^W(y)$ und $N_b^W(y)$ akzeptieren. Widerspruch zur Wahl von N_j^W.

In Stufe i ist also E durch Fall (ii) oder (iii) des Lemmas entstanden. Wieder über Argumentation der Laufzeit folgt, dass T_k keine Reduktion von $(X'_{a,b}(W), Y'_{a,b}(W))$ auf $(L(N_a^W), L(N_b^W))$ realisiert.

Zu (ii): Für $i = p^s_{a,b}$ werden höchstens dann Wörter der Länge $e(i)$ eingesetzt, wenn in Stufe i Fall (4) mit gefährdetem $d_{a,b,k}$, k beliebig, eintritt. Da E durch Lemma 25 entstanden ist, gilt $|E| \leq 2$.

Sollte Fall (4a) eingetreten, sein, ist $W^{=e(i)} \subsetneq E$, also $|W^{=e(i)}| \leq 1$. Sollte Fall (4b) eingetreten sein, ist nach Wahl von N_a, N_b die Erweiterung E durch Fall (ii) oder (iii) des Lemma 25 entstanden, also $|W^{=e(i)}| = 1$.

Also kann nicht gleichzeitig ein gerades, als auch ein ungerades Wort der Länge $e(i)$ in W liegen. Daher $(X'_{a,b}(W), Y'_{a,b}(W)) \in \mathrm{DisjNP}^W$ wie bereits argumentiert.

Zu (iii): Angenommen, es existiert ein vollständiges Paar $(L(N_a^W), L(N_b^W))$ für DisjNP^W. Nach (ii) dann $(X'_{a,b}(W), Y'_{a,b}(W)) \leq_m^{pp} (L(N_a^W), L(N_b^W))$.

Widerspruch zu (i), kann kein berechenbarer Transduktor existieren, der die Reduktion realisiert. □

Die verbleibende zu zeigende Eigenschaft (iii) von W folgt aus voriger Behauptung. Das schließt den Beweis zu Satz 26 ab. □

5.2 Verallgemeinerung der Konstruktion

Die Ergänzungen im vorigen Abschnitt demonstrierten, dass mit nur wenigen Änderungen das Verfahren auf andere Promise-Klassen angepasst werden kann. Unmittelbar stellt sich hierbei die Frage, welche Promises für das Verfahren ›schwach‹ genug sind, welche zu ›stark‹ sind.

Hierfür definieren wir uns zunächst über die Klasse $\mathrm{U}_p\mathrm{P}$ die Verallgemeinerung von UP, die Menge an Sprachen, die sich mit höchstens $p(|x|)$ akzeptierenden Pfaden abhängig Eingabelänge $|x|$ erkennen lassen.

Definition 29. [Bei89] Für totales $p\colon \mathbb{N} \to \mathbb{N}$ ist die Klasse $\mathrm{U}_p\mathrm{P}$ definiert als die Menge aller Sprachen, die von nichtdeterministischen Maschinen erkannt werden kann, welche für jede Eingabe der Länge n höchstens auf $p(n)$ vielen Pfaden akzeptiert.

$$\mathrm{U}_p\mathrm{P} =_{\mathrm{df}} \{L(N_i) \mid i \in \mathbb{N},\ \ \forall x, \mathrm{acc}_{N_i}(x) \leq p(|x|)\}.$$

Schon in Abschnitt 4.2 haben wir zwei Voraussetzungen identifiziert, welche zusammen für das Gelingen der P=NP∩coNP-Eigenschaft hinreichend sind; erstere werden wir im Folgenden abschwächen.

(C1) Für das abschließende Orakel O dürfen in allen Lücken $e(\cdot)$ höchstens konstant viele Wörter stehen. Ansonsten wird der in Behauptung 22 angegebene Polynomialzeit-Algorithmus im Allgemeinen *nicht* in der Zeitschranke terminieren, ist die Anzahl der Iterationen dann unbegrenzt. So galt in den bisherigen Konstruktionen, dass $|O \cap \Sigma^{e(i)}| \leq 2$ für alle i.

(C2) Mit O komplementär akzeptierende Maschinenpaare müssen zulassen, dass sich die relevanten Orakelwörter in den Stufen bestimmen lassen können. Hierfür muss sich zu jedem Zeitpunkt im Algorithmus ein akzeptierender Rechenweg anbieten. (Vgl. $(*)$ vom vorigen Kapitel.)

Hieraus ergibt sich in der Konstruktion die Konsequenz, dass ›bevorzugt‹ nicht alle Wörter in die Lücken eingesetzt werden, damit $\mathrm{NP} \cap \mathrm{coNP}$-Maschinenpaare endgültig nicht mehr komplementär akzeptieren. (Das heißt Vorgaben $c_{a,b}$ erfüllt.)

Wird also in Stufe i versucht eine Vorgabe $r \neq c_{a,b}$ mit Erweiterung $E \subseteq \Sigma^{e(i)}$ für die Lücke $e(i)$ zu erfüllen, ist nicht sichergestellt, dass $E \neq \emptyset$ auch vollständig in die Lücke eingesetzt wird. Insbesondere kann geschehen, dass $O \cap \Sigma^{e(i)} \subsetneq E$.

Aus Eigenschaft (C1) lässt sich schon unmittelbar beobachten, dass sich die in Satz 17 angegebene Konstruktion auf mehr als einen, aber konstant viele akzeptierende Pfade verallgemeinern lässt. Insbesondere existiert für $c \in \mathbb{N}$, $c(|x|) = c$ ein Orakel, relativ zu diesem $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ und $\mathrm{U}_c\mathrm{P}$ keine $\leq^{\mathrm{p}}_{\mathrm{m}}$-vollständigen Mengen enthält: Einerseits lässt sich Lemma 16 auf $\mathrm{U}_c\mathrm{P}$ verallgemeinern, andererseits terminiert der in Behauptung 22 angegebene Algorithmus nach höchstens $c + 1$ vielen, also konstant vielen, Iterationen.

Gleichzeitig fällt mit Rückblick auf den Aufbau des in Behauptung 22 angegeben Algorithmus auf, dass die in (C1) angegebene Schranke intuitiv zu streng ist. Die polynomielle Laufzeitschranke sollte in der Lage sein, auch polynomiell viele Iterationen im Algorithmus zuzulassen. Damit wären auch polynomiell viele Wörter in den Lücken $e(\cdot)$ möglich.

Behauptung 30. *Sei p ein Polynom. Der Algorithmus von Behauptung 22 in der Konstruktion von W terminiert auch dann in Polynomialzeit, wenn $|W \cap \Sigma^{e(i)}| \leq p(e(i))$ für alle i.*

Beweis. Seien N_a, N_b die beiden im Algorithmus angegebenen Maschinen, q die gemeinsame polynomielle Laufzeitschranke der Maschinen.

Wir zeigen, dass für alle $|x|$ mit $e(i_0) < \log |x|$ der Algorithmus nach höchstens $p(q(|x|))$ vielen Iterationen terminiert. Damit terminiert der Algorithmus in einer polynomiellen Laufzeitschranke in Abhängigkeit der Eingabelänge $|x|$. Sei i wie in Schritt (1) berechnet.

Ist $q(|x|) < e(i)$, dann wird weder $N_a(x)$ noch $N_b(x)$ Orakelfragen der Länge $e(i)$ stellen. Der Algorithmus terminiert dann schon nach der ersten Iteration in (4a), ist ja dann $E' = Q^{=e(i)} = \emptyset$.

Ist im anderen Fall $e(i) \leq q(|x|)$, dann $|W^{=e(i)}| \leq p(e(i)) \leq p(q(|x|))$. Nach höchstens $p(q(|x|))$ Iterationen ist dann $E' = W^{=e(i)}$, und der Algorithmus terminiert. □

Mit diesem Ergebnis können wir die ursprüngliche, zu starke Voraussetzung (C1) abschwächen:

(C1′) Für das abschließende Orakel O muss ein beliebiges, aber *festes* Polynom p existieren, sodass $|O \cap \Sigma^{e(i)}| \leq p(e(i))$ für alle i gilt.

Unmittelbar bietet sich nun an, in der Konstruktion den UP-Promise auf polynomiell viele akzeptierende Rechenwege auszudehnen. Im Folgenden werden wir deshalb die Konstruktion so verändern, dass einerseits wie bisher relativiert $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$, und gleichzeitig für gegebenes nicht-konstantes Polynom p die Klasse $\mathrm{U}_p\mathrm{P}$ von der größeren Promise-Klasse $\mathrm{U}_{p+1}\mathrm{P}$ getrennt wird.

Vollständige Mengen, wie die Hypothese UP behandelt, werden hier nicht betrachtet. Wir betrachten Reduktionen und vollständige Mengen erst bei der größeren Promise-Klasse FewP. Insbesondere zeigt sich mit nicht-konstantem p, dass $\leq^{\mathrm{p}}_{\mathrm{m}}$-vollständige Mengen für $\mathrm{U}_p\mathrm{P}$ genau dann existieren, wenn für FewP $\leq^{\mathrm{p}}_{\mathrm{m}}$-vollständige Mengen existieren. Deshalb wird die Frage, ob sich auch vollständige Mengen für $\mathrm{U}_p\mathrm{P}$ ausschließen lassen, erst im nächsten Abschnitt diskutiert.

Nichtsdestotrotz ist es möglich, für alle $\mathrm{U}_p\mathrm{P}$-Maschinen das korrekte Erkennen einer Zeugensprache auszuschließen, bzw. den Promise zu zerstören. Darauf aufbauend werden die Änderungen am Verfahren angegeben, sodass ein Orakel mit den behaupteten Eigenschaften konstruiert wird. Wir definieren uns die Zeugensprache $D'(W) = \{1^n \mid n = e(i), \exists w : |w| = n \wedge w \in W\}$.

Lemma 31. *Sei $p > 0$ ein beliebiges Polynom, $j \in \mathbb{N}$ beliebig, $n =_{\mathrm{df}} e(i)$ hinreichend groß sodass $2^n > 4\,(p(n))(t_j(n) + 1)$, und $C \subseteq \Sigma^*$ mit keinen Wörtern der Länge n in C.*

Es existiert eine Erweiterung $E \subseteq \Sigma^n$ sodass einer der folgenden Fälle eintritt:

(i) $|E| = p(n) + 1$ und $N_j^{C \cup E}$ akzeptiert 1^n auf mehr als $p(n)$ Rechenwegen.

(ii) $|E| \leq 1$ mit $1^n \in D'(C \cup E) \Leftrightarrow 1^n \notin L(N_j^{C \cup E})$.

Beweis. Zunächst erfolgt die Argumentation ähnlich wie in Lemma 8. Angenommen, keine Erweiterung E existiert sodass Fall (i) oder Fall (ii) eintritt. Dann ist $1^n \notin L(N_j^C)$, und für alle $w \in \Sigma^n$ auch $1^n \in L(N_j^{C \cup \{w\}})$.

Für alle $w \in \Sigma^n$ definieren wir wieder $Q(w)$ als die Menge aller Orakelfragen auf dem Annahmepfad von $N_j^{C \cup \{w\}}(1^n)$. Wieder $w \in Q(w)$.

Wir definieren nun den Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ mit

$$\mathcal{V} =_{\mathrm{df}} \Sigma^n, \quad \mathcal{E} =_{\mathrm{df}} \{(u, v) \mid v \in Q(u)\}.$$

Es ist $|\mathcal{V}| = 2^n =_{\mathrm{df}} m$, und der Außengrad jedes Knotens auf $\leq t_j(n) =_{\mathrm{df}} d$ beschränkt. Im Gegensatz zu den bisherigen Anwendungen des kombinatorischen Lemmas 7 setzen wir nun $\alpha =_{\mathrm{df}} p(n) + 1$, nichtsdestotrotz ist nach Wahl von n die Voraussetzung des Lemmas erfüllt:

$$m = 2^n > 4\,(p(n))(2d + 1) > (p(n) + 1) \cdot 2(2d + 1) = \alpha \cdot 2(d + 1).$$

Damit existiert eine unabhängige Menge $\mathcal{U} \subseteq \mathcal{V}$ der Größe $p(n) + 1$ in $\mathcal{G}$, also hat der induzierte Teilgraph $\mathcal{G}[\mathcal{U}]$ keine Kanten zwischen paarweise verschiedenen Knoten.

Wir zeigen nun analog zum Beweis in Lemma 8, dass $N_j(1^n)$ mit Orakel $C \cup \mathcal{U}$ auf mehr als $p(n)$ Rechenwegen akzeptieren wird: Für jedes $u \in \mathcal{U}$ existiert nach Annahme ein akzeptierender Rechenweg $N_j^{C \cup \{u\}}(1^n)$. Die (jeweils lexikalisch kleinsten) Rechenwege sind alle paarweise verschieden, ist ja für alle paarweise verschiedenen $a, b \in \mathcal{U}$ auch

$a \in Q(a)$, $a \notin Q(b)$, also $Q(a) \neq Q(b)$. Weiterhin bleibt für jedes $u \in \mathcal{U}$ der Rechenweg identisch vorhanden in der Berechnung $N_j^{C \cup \mathcal{U}}(1^n)$, ist ja $\mathcal{U} \cap Q(u) = \{u\}$.

Es existieren daher $|\mathcal{U}| = p(n) + 1 > p(n)$ viele akzeptierende Rechenwege auf der Berechnung $N_j^{C \cup \mathcal{U}}(1^n)$. Mit Wahl $E =_{\text{df}} \mathcal{U} \subseteq \Sigma^n$ ist dann widersprüchlicherweise Fall (i) eingetreten. □

Satz 32. *Sei $p > 0$ ein Polynom. Es existiert ein Orakel relativ zu diesem*

(i) $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$*, und*

(ii) $\mathrm{U}_{p(n)}\mathrm{P} \subsetneq \mathrm{U}_{p(n)+1}\mathrm{P}$.

Beweis. Wieder geben wir nur die Änderungen an der bisherigen Konstruktion aus Satz 17 an.

Zunächst verändern wir die Menge an Vorgaben, und wählen

$$\mathcal{R} =_{\text{df}} \{c_{a,b} \mid a, b \in \mathbb{N}, a \neq b\} \cup \{t_j \mid j \in \mathbb{N}\} = \{c_{(0)}, c_{(1)}, \ldots, t_0, t_1, \ldots\},$$

und ordnen diese wieder isomorph zu $(\mathbb{N}, <)$ folgendermaßen an: $c_{(0)} \prec t_0 \prec c_{(1)} \prec t_1 \prec \cdots$. Im Vergleich zur ursprünglichen Konstruktion lassen wir also die Angabe von Reduktionsfunktionen in den t-Vorgaben aus.

Wir nennen t_j in Stufe i gefährdet, wenn $2^{e(i)} > 4\,(p(e(i)))(t_j(e(i)) + 1)$ und $t_j(e(i)) < e(i+1)$. Das enspricht der Voraussetzung aus vorigem Lemma mit $n = e(i)$. Formal ändert sich die in Satz 17 angegebene Vorschrift, wenn die kleinste gefährdete Vorgabe $r = t_j$ ist. Fall (3) lautet dann:

(3) Ist $r = t_j$ die kleinste gefährdete Vorgabe, bestimme $E \subseteq \Sigma^{e(i)}$ aus vorigem Lemma (mit W_{i-1} als Wahl für C). Wähle unter allen Teilmengen $E' \subsetneq E$ jene aus, für die das durch E' gefährdete $c_{a,b} \prec d_{a,b,k}$ minimal ist.

  (a) Existiert ein solches E', setze $W_i =_{\text{df}} W_{i-1} \cup E'$ und markiere $c_{a,b}$ als erfüllt.

  (b) Andernfalls, wenn keine solche echte Teilmenge existiert, setze $W_i =_{\text{df}} W_{i-1} \cup E$, und markiere t_j als erfüllt.

Behauptung 18 gilt analog, alle t_j sind erfüllt. Wir zeigen damit Eigenschaft (ii) des Orakels.

Behauptung 33. *Es ist $D'(W) \in \mathrm{U}_{p(n)+1}\mathrm{P} \setminus \mathrm{U}_{p(n)}\mathrm{P}$.*

Beweis. Da in allen Konstruktionsschritten i höchstens $p(e(i))+1$ Wörter der Länge $e(i)$ hinzugefügt wurden, existiert analog zur Beobachtung 6(iii) eine nichtdeterministische Maschine, die auf höchstens $p(|x|) + 1$ Rechenwegen entscheiden kann, ob $x \in D'(W)$. Damit $D'(W) \in \mathrm{U}_{p(n)+1}\mathrm{P}$.

Angenommen, $D'(W) \in \mathrm{U}_{p(n)}\mathrm{P}$. Ähnlich wie im Beweis von Behauptung 14(i) existiert ein $j \in \mathbb{N}$, sodass $L(N_j^W) = D'(W)$ und $N_j^W(x)$ auf höchstens $p(|x|)$ Rechenwegen akzeptiert. Sei nun i die Stufe, in der t_j erfüllt ist; in dieser Stufe ist Fall (3b) eingetreten.

Wenn E durch Fall (i) des vorigen Lemmas entstanden ist, akzeptiert $N_j^{W_i}(1^{e(i)})$ auf mehr als $p(e(i))$ Rechenwegen. Nachdem t_j in Stufe i gefährdet war, galt auch $t_j(e(i)) < e(i{+}1)$. Damit werden bei Berechnung $N_j^{W}(1^{e(i)})$ die Orakelfragen identisch beantwortet, also akzeptiert auch $N_j^{W}(1^{e(i)})$ auf mehr als $p(e(i))$ Rechenwegen; Widerspruch zur Wahl von N_j.

Ansonsten, wenn E durch Fall (ii) entstanden ist, akzeptiert $N_j^{W_i}(1^{e(i)}) \Leftrightarrow 1^{e(i)} \notin D'(W_i)$. Nach gleicher Argumentation über die Laufzeit ist dann $1^{e(i)} \in L(N_j^{W}) \Leftrightarrow 1^{e(i)} \notin D'(W)$; Widerspruch zur Wahl von N_j.

Also war die Annahme falsch, $D'(W) \notin \mathrm{U}_{p(n)}\mathrm{P}$. □

Behauptungen 20–22 gelten analog, insbesondere sind für alle i höchstens $p(e(i)) + 1$ Wörter der Länge $e(i)$ im Orakel. Nach Behauptung 30 hält damit der in der Konstruktion angegebene Algorithmus die geforderte polynomielle Laufzeitschranke ein. Damit gilt auch Eigenschaft (i) des Orakels. □

5.3 Grenzen des Verfahrens

Fest polynomiell viele Wörter Eine unmittelbare Grenze des Verfahren ergibt sich schon beim leicht stärkeren Promise der Klasse FewP.

$$\begin{aligned}\mathrm{FewP} &=_{\mathrm{df}} \{L(N_i) \mid i \in \mathbb{N},\ \text{es ex. Polynom } p \text{ sodass } \forall x : \mathrm{acc}_{N_i}(x) \leq p(|x|)\} \\ &= \bigcup_{c \geq 0} \mathrm{U}_{(n^c+c)}\mathrm{P} = \{L(N_i) \mid i \in \mathbb{N}, \forall x : \mathrm{acc}_{N_i}(x) \leq |x|^i + i\}. \quad \text{[Bei89]}\end{aligned}$$

Für die untersuchten Promise-Klassen gelten also die Inklusionen

$$\mathrm{UP} \subseteq \mathrm{U}_{t_1}\mathrm{P} \subseteq \mathrm{U}_{t_2}\mathrm{P} \subseteq \cdots \subseteq \mathrm{FewP} \subseteq \mathrm{NP}.$$

Insbesondere kann FewP als nächstgrößere relevante Promise-Klasse verstanden werden. Die Klassen FewP und UP wurden deutlich intensiver untersucht als $\mathrm{U}_p\mathrm{P}$, gerade auch weil die beiden Klassen mit Einwegfunktionen zusammenhängen. [vgl. CH90, S. 96] Weiterhin ist FewP nach UP die nächstgrößere Promise-Klasse, die unter $\leq_{\mathrm{m}}^{\mathrm{p}}$-Reduktionen abgeschlossen ist. Über ein Padding-Argument lässt sich beobachten, dass zu gegebenem nicht-konstanten Polynom p entweder $\mathrm{U}_p\mathrm{P} = \mathrm{FewP}$, oder $\mathrm{U}_p\mathrm{P}$ nicht unter $\leq_{\mathrm{m}}^{\mathrm{p}}$-Reduktionen abgeschlossen ist.

Wie angekündigt zeigt folgende Beobachtung durch das gleiche Argument, dass der Vollständigkeitsbegriff zwischen den Klassen $\mathrm{U}_p\mathrm{P}$ und FewP identisch ist.

Beobachtung 34. *Sei p ein beliebiges nicht-konstantes Polynom. Folgende Aussagen sind äquivalent:*

(i) Es existieren $\leq_{\mathrm{m}}^{\mathrm{p}}$-vollständige Mengen für $\mathrm{U}_p\mathrm{P}$.

(ii) Es existieren $\leq_{\mathrm{m}}^{\mathrm{p}}$-vollständige Mengen für FewP.

Beweis. Es gilt (i) ⇒ (ii): Sei nach Voraussetzung B eine für $\mathrm{U}_p\mathrm{P}$ vollständige Menge, und $A \in \mathrm{FewP}$ beliebig. Wir geben eine Menge $A' \in \mathrm{U}_p\mathrm{P}$ an, sodass $A \leq^{\mathrm{p}}_{\mathrm{m}} A'$. Sei N die nichtdeterministische Maschine, die A entscheidet, insbesondere ist nach Definition für geeignetes Polynom q dann $\mathrm{acc}_N(x) \leq q(|x|)$. Wir definieren

$$A' = \{0^{q(|w|)}1w \mid N(w) \text{ akzeptiert}\}.$$

Es ist $A' \in \mathrm{U}_p\mathrm{P}$ gerade weil $\mathrm{acc}_N(w) \leq q(|w|) \leq |x| \leq p(|x|)$. Weiterhin ist $A \leq^{\mathrm{p}}_{\mathrm{m}} A'$ über Reduktionsfunktion $f\colon w \mapsto 0^{q(|w|)}1w$, denn

$$w \in A \;\Leftrightarrow\; N(w) \text{ akz.} \;\Leftrightarrow\; 0^{q(|w|)}1w \in A' \;\Leftrightarrow\; f(w) \in A'.$$

Da nach Vollständigkeit $A \leq^{\mathrm{p}}_{\mathrm{m}} A' \leq^{\mathrm{p}}_{\mathrm{m}} B$ ist B vollständig für FewP.

Es gilt (ii) ⇒ (i): Sei nun B eine für FewP vollständige Menge, und $A \in \mathrm{U}_p\mathrm{P}$ beliebig. Wir geben analog eine Menge $B' \in \mathrm{U}_p\mathrm{P}$ an, sodass $B \leq^{\mathrm{p}}_{\mathrm{m}} B'$. Sei wieder N eine nichtdeterministische Maschine, sodass $L(N) = B$, $\mathrm{acc}_N(x) \leq q(|x|)$. Wir definieren $B' = \{0^{q(|w|)}1w \mid N(w) \text{ akz.}\}$, dann ist $\mathrm{acc}_N(w) \leq q(|w|) \leq |x| \leq p(|x)$, also $B' \in \mathrm{U}_p\mathrm{P}$. Analog ist $B \leq^{\mathrm{p}}_{\mathrm{m}} B'$ über Reduktionsfunktion $w \mapsto 0^{q(|w|)}1w$. Da $A \in \mathrm{FewP}$ ist nach Vollständigkeit $A \leq^{\mathrm{p}}_{\mathrm{m}} B \leq^{\mathrm{p}}_{\mathrm{m}} B'$, also B' vollständig für $\mathrm{U}_p\mathrm{P}$. □

Die im vorigen Abschnitt aufgeschobene Frage nach vollständigen Mengen in $\mathrm{U}_p\mathrm{P}$ werden wir deshalb aus Perspektive der Klasse FewP betrachten. Nichtsdestotrotz eignet sich das hier vorgestellt Verfahen nicht einmal, irgendwelche relativierten Aussagen bezüglich dieser Klasse zu erreichen. Ein analoges Ergebnis wie Satz 17, welches vollständige Mengen für FewP ausschließt, scheint damit auch nicht möglich.

Die Definition des Promises suggeriert zwar, dass höchstens polynomiell große Erweiterungen notwendig sind, dennoch müssen bei der Konstruktion aber *alle* möglichen Polynome bearbeitet werden. Die von (C1′) geforderte *feste* polynomielle Schranke ist damit nicht mehr gesichert. Damit ist die Konstruktion nicht mehr auf triviale Weise möglich.

Beispielsweise ist eine relativierte Trennung von FewP und NP schon unmöglich. Wolle man Orakel W definieren, sodass

$$D'(W) = \{1^n \mid n = e(i), \exists w : |w| = n \wedge w \in W\} \in \mathrm{NP} \setminus \mathrm{FewP}$$

gilt, ist sofort klar, dass kein festes Polynom p die Ungleichung $|W \cap \Sigma^{e(i)}| \leq p(e(i))$ sichert, wäre ansonsten $D'(W) \in \mathrm{U}_p\mathrm{P} \subseteq \mathrm{FewP}$. Die von Bedingung (C1′) für die Konstruktion geforderte Schranke ist damit nicht möglich, damit wäre $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ nicht mehr gesichert. Stärkere Ergebnisse bezüglich der Nichtexistenz vollständiger Mengen in FewP scheinen durch dieses Verfahren also auch ausgeschlossen.

Robuste Zeugensprachen Pudlák untersucht in seiner bereits genannten Arbeit auch die Hypothese DisjcoNP. Analog zu DisjNP ist DisjcoNP die Hypothese dass DisjcoNP

keine $\leq_m^{pp}$-vollständigen Paare enthält. Das Paar (A, B) ist, analog zu Definition 23, Element der Klasse DisjcoNP^O wenn $A, B \in \text{coNP}^O$, und $A \cap B = \emptyset$. Die $\leq_m^{pp}$-Reduktionen zwischen DisjcoNP-Paaren sind identisch definiert, wieder können wir uns auf Reduktionsfunktionen ohne Orakelzugriff beschränken.

Die Hypothese DisjcoNP ist insofern relevant, als da jene Hypothese bzw. SAT als symmetrisches Analog zu DisjNP bzw. CON verstanden werden kann. Im Gegensatz zu CON sagt die Hypothese SAT aus, dass keine p-optimalen Beweissysteme existieren, die Erfüllbarkeit beweisen. (Im Gegensatz zu Gültigkeit bei CON.) Insbesondere impliziert jede der Hypothesen DisjNP, DisjcoNP, NP∩coNP die kombinierte Hypothese CON ∨ SAT. [vgl. Pud17; vgl. BKM09]

Daraus kann argumentiert werden, dass die Hypothese DisjcoNP zu NP∩coNP in der gleichen Beziehung wie DisjNP zu NP∩coNP stehen könnte. In Abschnitt 5.1 wurde schon gezeigt, dass relativ zu einem Orakel DisjNP $\not\Rightarrow$ NP∩coNP. Es eröffnet sich die Frage, ob auch eine relativierbare symmetrische Implikation widerlegt werden kann; das heißt ob ein Orakel anzugeben ist, relativ zu dem DisjcoNP $\not\Rightarrow$ NP∩coNP. (Vgl. auch mit Abbildung 1.)

Aber auch der DisjcoNP-Promise ist für das hier vorgestellte Verfahren vermutlich schon zu stark. Ohne größeren Aufwand ließe sich die hier angegebene Konstruktion nicht erweitern, um ein Orakel anzugeben, relativ zu dem DisjcoNP ∧ ¬NP∩coNP gilt. Die erste Schwierigkeit ergibt sich schon daraus, ›robuste‹ Definitionen für das allgemeine Zeugenpaar $(X''_{a,b}(O), Y''_{a,b}(O)) \in \text{DisjcoNP}^O$ anzugeben.

Wie wir in Bedingung (C2) festgestellt haben, treten Situationen in der Konstruktion auf, in der *nicht die gesamte* Erweiterung E in die Lücke $e(i)$ in das Orakel eingesetzt wurde. Insbesondere können wir uns dann für Promise-erfüllende Maschinen N_a^O, N_b^O mit $(\overline{L(N_a^O)}, \overline{L(N_b^O)}) \in \text{DisjcoNP}^O$ nicht einmal verlassen, dass $O \cap \Sigma^{e(i)} \neq \emptyset$ für alle Potenzen $i = p_{a,b}^s$.

Wird also in der Konstruktion eine Lücke $e(p_{a,b}^s)$ in O freigelassen, muss das Zeugenpaar ›hinreichend robust‹ sein, sodass immer noch $(X''_{a,b}(O), Y''_{a,b}(O)) \in \text{DisjcoNP}^O$. Andernfalls eignet sich das Zeugenpaar nicht dafür, um die Vollständigkeit von $(L(N_a^O), L(N_b^O))$ zu widerlegen. Damit sind einfache Zeugensprachen wie

$$X''_{a,b}(O) =_{\text{df}} \{1^n \mid n = e(p_{a,b}^s), \forall z : |z| = n \wedge z \text{ gerade} \to z \notin O\},$$
$$Y''_{a,b}(O) =_{\text{df}} \{1^n \mid n = e(p_{a,b}^s), \forall z : |z| = n \wedge z \text{ ungerade} \to z \notin O\},$$

analog zur im zweiten Kapitel definierte coNP-Zeugensprache Y insbesondere nicht geeignet: Sei dafür $n = e(p_{a,b}^s)$, $O \cap \Sigma^n = \emptyset$, dann ist offenbar $1^n \in X''_{a,b}(O) \cap Y''_{a,b}(O)$.

Schon durch diese Bedingung ist die Wahl der Zeugensprachen eingeschränkt. Sei im Folgenden N_a^O, N_b^O mit $\overline{L(N_a^O)} \cap \overline{L(N_b^O)} = \emptyset$ ein Promise-erfüllendes Maschinenpaar. Sinnvollerweise wünschen wir wieder, dass die Zugehörigkeit $w \in X''_{a,b}(O)$ bzw. $w \in Y''_{a,b}(O)$ von Wörtern $w \in \Sigma^{e(i)}$ nur von den Orakelwörtern in der Lücke $e(i)$ abhängt. Ist

also kein Wort der Länge $n = e(p^s_{a,b})$ im Orakel O vorhanden, müssen die Zeugensprachen auf allen $w \in \Sigma^n$ disjunkt sein.

Die Wahl eines für das Verfahren günstige Paar an Zeugensprachen ist dem Autor nicht gelungen. Zwar lassen sich Zeugensprachen definieren, die den bisherigen Bedingungen genügen, eine weitere Schwierigkeit besteht aber immer noch darin, eine geeignetes Verfahren zur Diagonalisierung anzugeben. Im Folgenden werden weitere Indizien angegeben, warum das Verfahren nicht hinreichend stark für das DisjcoNP-Promise ist.

Sei im Folgenden $C \subseteq \Sigma^*$ analog zu Lemmata 9 bzw. 25 (für NP $\cap$ coNP bzw. DisjNP) als ›unfertiges‹ Orakel ohne Wörter in der Lücke $n = e(i)$, $i = p^s_{a,b}$ gegeben. Das Ziel besteht darin, eine Belegung $E \subseteq \Sigma^n$ der Lücke zu definieren sodass die von T_k durchgeführte Reduktion von $(X''_{a,b}(C \cup E), Y''_{a,b}(C \cup E))$ auf die vom Maschinenpaar N_a, N_b erkannte Sprache nicht korrekt ist. Falls das nicht möglich sein sollte, ist E so zu wählen dass die beiden Maschinen nicht mehr disjunkt ablehnen.

Nach voriger Überlegung existiert notwendigerweise kein $w \in \Sigma^n$ mit $w \in X''_{a,b}(C)$ und $w \in Y''_{a,b}(C)$. Des Weiteren können wir für geeignete Zeugensprachen annehmen, dass $X''_{a,b}(C)$ und $Y''_{a,b}(C)$ nicht die Menge Σ^n partitionieren: Angenommen

$$\forall w \in \Sigma^n : w \in X''_{a,b}(C) \vee w \in Y''_{a,b}(C),$$

dann ist eine durch T_k korrekt durchgeführte Reduktion nicht zu zerstören. Durch Entfernen eines Wortes aus den Zeugensprachen durch Wahl von E, also sodass zum Beispiel $w \notin X''_{a,b}(C \cup E)$, wird nur der ›Definitionsbereich‹ der Promise-Reduktion verkleinert; auf diese Weise bleibt für alle $E \subseteq \Sigma^n$ die Reduktion korrekt.

Ohne Beschränkung können wir also ausgehen, dass für die Sprachen

$$X''_{a,b}(C) \cap \Sigma^n = \emptyset, \;\; Y''_{a,b}(C) \cap \Sigma^n = \emptyset \text{ gilt.}$$

Weiterhin müssen, wie eben argumentiert, die Zeugensprachen zulassen, dass durch entsprechende Wahl von E die Zugehörigkeit eines Wortes zur entsprechenden Sprache zu bestimmen ist. Eine Erweiterung E ist *konsistent*, wenn dadurch das Paar disjunkt bleibt, also wenn

$$\forall w \in \Sigma^n : w \notin X''_{a,b}(C \cup E) \vee w \notin Y''_{a,b}(C \cup E).$$

Optimalerweise existiert nun eine konsistente Erweiterung, die eine korrekte Reduktion ausschließt, also sodass $w \in X''_{a,b}(C \cup E)$ und $T_k(w) \notin \overline{L(N_a^{C \cup E})}$, oder analog für $Y''_{a,b}$, N_b. Dennoch könnte im Zweifelsfall für alle $w \in \Sigma^n$ und alle konsistenten Erweiterungen E die Reduktion korrekt durchgeführt werden, beispielsweise wenn N_a^O die Zeugensprache $\overline{X''_{a,b}(O)}$, und N_b^O die Zeugensprache $\overline{Y''_{a,b}(O)}$ entscheidet. Verfahren wir wie in der DisjNP-Diagonalisierung, können wir durch Wahl eines ausgezeichneten w, zwei konsistenten Erweiterungen E_X, E_Y

$$\begin{aligned} &w \in X''_{a,b}(C \cup E_X), \text{ also nach Annahme } T_k(w) \in \overline{L(N_a^{C \cup E_X})}, \quad \text{und} \\ &w \in Y''_{a,b}(C \cup E_Y), \text{ also nach Annahme } T_k(w) \in \overline{L(N_b^{C \cup E_Y})} \end{aligned}$$

erreichen. Im Gegensatz zur analogen DisjNP-Diagonalisierung ist nun aber nicht klar, wie konsistente E_X und E_Y zu (ggf. inkonsistentem) E vereinigt werden können, um den Promise der beiden Maschinen zu zerstören. Die *ablehnenden* Pfade der Maschinen können nicht trivial fixiert werden, sodass $T_k(w) \in \overline{L(N_a^{C \cup E})} \cap \overline{L(N_b^{C \cup E})}$. Dieser essenzielle Unterschied – ablehnende Pfade fixieren, und keine akzeptierende Pfade – stellt vermutlich das größte Indiz da, weshalb sich das Verfahren nicht für die Hypothese DisjcoNP eignet.

6 Fazit

Die Arbeit erzielt als primäres Ergebnis den Nachweis der paarweisen Unabhängigkeit zwischen den relativierten Versionen Pudláks Hypothesen UP und NP∩coNP. Dose und Glaßer zeigten mit einem Orakel bereits, dass die Implikation NP∩coNP ⇒ UP nicht relativierbar hält. Die in Kapitel 4 vorgestellte Konstruktion gibt dann ein analoges Orakel für die konverse Implikation an.

Das in Kapitel 3 vorgestellte Orakel, sowie das bekannte P=NP-Orakel von Baker et al., vervollständigen das Bild und geben analoge Orakel an, welche die relativierbare Implikation UP ⇒ ¬NP∩coNP bzw. dessen Konverse ausschließen. Wir erreichen das in Tabelle 1 dargestellte Bild, sowie folgendes Korollar:

Korollar 35. *Die Hypothesen* UP *und* NP∩coNP *sind paarweise unabhängig unter relativierbaren Beweisen. Für jede Wahl von* $\mathsf{A} \in \{\mathsf{UP}, \neg\mathsf{UP}\}$*, und jede Wahl* $\mathsf{B} \in \{\mathsf{NP}\cap\mathsf{coNP}, \neg\mathsf{NP}\cap\mathsf{coNP}\}$ *existiert ein Orakel, sodass relativiert* $\mathsf{A} \not\Rightarrow \mathsf{B}$*.*

Hiermit wurde ein weiteres Indiz geliefert, dass ein Nachweis oder Widerlegung von Implikationen zwischen den beiden angesprochenen Hypothesen nur mit anspruchsvollsten nichtrelativierbaren Beweismethoden erfolgen kann.

Die Konstruktion von W in Satz 17 verwendete dabei das Verfahren von Baker et al. als Grundlage. Darauf aufbauend wurde in Kapitel 5 als sekundäres Ziel die prinzipielle Möglichkeit des Verfahrens untersucht. Es wurde beobachtet, dass sich die zu UP verwandte Hypothese DisjNP auch in die Konstruktion integrieren ließ. Unter relativierbaren Beweisen wird man nicht NP∩coNP als Konsequenz der möglicherweise stärkeren Hypothese UP ∧ DisjNP beweisen können.

Korollar 36. *Es existiert ein Orakel für das die relativierte Hypothese* UP ∧ DisjNP $\not\Rightarrow$ NP∩coNP *gilt.*

Nichtsdestotrotz wurden in Abschnitt 5.3 mögliche Schranken des Verfahrens identifiziert: Ein analoges Resultat für die nächstgrößere signifikante Promise-Klasse FewP

UP	NP∩coNP	Orakel nachgewiesen durch
+	+	Satz 10
+	−	Satz 17
−	+	Dose und Glaßer [DG19, Cor. 4.2]
−	−	Baker, Gill und Solovay [BGS75, Thm. 1] (Zusammenfallen von P und NP)

Tabelle 1: In jeder Zeile wird ein Orakel nachgewiesen, für welches die Hypothesen in den beiden linken Spalten gleichzeitig zutreffen. Ein + bezeichnet die relativierte Hypothese im Spaltenkopf, ein − die relativierte Negation der Hypothese.

lässt sich mit dem hier vorgestellten Verfahren nicht, bzw. nicht so einfach wie im Fall von UP bzw. DisjNP zeigen. Insbesondere ließ sich kein Orakel konstruieren, relativ zu diesem FewP keine vollständigen Sprachen besitzt, aber $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$.

Auch für die Hypothese DisjcoNP ließ sich das Verfahren nicht anpassen. Es konnte kein Orakel konstruiert werden, relativ zu diesem DisjcoNP $\not\Rightarrow$ NP$\cap$coNP gilt. Aus den Vorgaben der Konstruktion war nicht ersichtlich, wie geeignete Zeugensprachen für DisjcoNP definiert werden können. Insbesondere müssen geeignete Zeugensprachen hinreichend ›robust‹ im Sinne des Verfahrens sein. Für diesen Fall wurden abschließend Indizien angegeben, dass durch die Wahl von günstigen Zeugensprachen es nicht mehr möglich erscheint, den DisjcoNP-Promise zu zerstören. Damit konnten nicht bei allen Maschinenpaaren die Vollständigkeit ausgeschlossen werden.

Einerseits kann diese Grenze als allgemeine Schwäche des Verfahrens interpretiert werden. Insbesondere ist die angezielte Eigenschaft $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ eigentlich eine stärkere Aussage als ¬NP$\cap$coNP. Andererseits könne man diese Schwierigkeit der Orakelkonstruktion möglicherweise als Indiz verstehen, dass die Hypothese DisjcoNP als mehr als nur das ›symmetrisches Analog‹ zu DisjNP verstanden werden kann. Nur wenig Anhaltspunkte sind bekannt, welche die beiden Hypothesen stützen. [Pud17, S. 439]

Es bleibt interessant, ob sich weitere Implikationen zwischen Pudláks Hypothesen zeigen lassen können, bzw. welche relativierbare Implikationen ausgeschlossen werden können. Insbesondere, welche zukünftigen Ergebnisse sich im Bezug auf die Implikation DisjcoNP ergeben, die zurzeit nur in wenigen Arbeiten untersucht wurde.

Literatur

[Bei89] Richard Beigel. »On the relativized power of additional accepting paths«. In: *Structure in Complexity Theory, Fourth Annual Conference*. IEEE Comput. Soc. Press, 1989, S. 216–224. DOI: 10.1109/SCT.1989.41827.

[BGS75] Theodore Baker, John Gill und Robert Solovay. »Relativizations of the P=? NP Question«. In: *SIAM Journal on Computing* 4.4 (1975), S. 431–442. DOI: 10.1137/0204037.

[BKM09] Olaf Beyersdorff, Johannes Köbler und Jochen Messner. »Nondeterministic functions and the existence of optimal proof systems«. In: *Theoretical Computer Science* 410.38 (2009), S. 3839–3855. DOI: 10.1016/j.tcs.2009.05.021.

[CH90] Jin-yi Cai und Lane A. Hemachandra. »On the power of parity polynomial time«. In: *Mathematical Systems Theory* 23.1 (1990), S. 95–106. DOI: 10.1007/BF02090768.

[DG19] Titus Dose und Christian Glaßer. »NP-Completeness, Proof Systems, and Disjoint NP-Pairs«. In: *Electronic Colloquium on Computational Complexity* 50 (2019). URL: https://eccc.weizmann.ac.il/report/2019/050/.

[Dos19] Titus Dose. »P-Optimal Proof Systems for Each Set in NP but no Complete Disjoint NP-Pairs Relative to an Oracle«. In: *ArXiv e-prints* (27. Juni 2019). arXiv: 1904.06175 [cs.CC]. URL: http://arxiv.org/abs/1904.06175.

[EY80] Shimon Even und Yacov Yacobi. »Cryptocomplexity and NP-completeness«. In: *Automata, Languages and Programming. Seventh Colloquium Noordwijkerhout, the Netherlands July 14–18, 1980*. Hrsg. von Jaco Bakker und Jan Leeuwen. Bd. 85. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1980, S. 195–207. DOI: 10.1007/3-540-10003-2_71.

[Gol06] Oded Goldreich. »On Promise Problems: A Survey«. In: *Theoretical Computer Science: Essays in Memory of Shimon Even*. Hrsg. von Oded Goldreich, Arnold L. Rosenberg und Alan L. Selman. Bd. 3895. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, S. 254–290. DOI: 10.1007/11685654_12.

[GS88] Joachim Grollmann und Alan L. Selman. »Complexity Measures for Public-Key Cryptosystems«. In: *SIAM Journal on Computing* 17.2 (1988), S. 309–335. ISSN: 0097-5397. DOI: 10.1137/0217018.

[HH86] Juris Hartmanis und Lane A. Hemachandra. »Complexity classes without machines: On complete languages for UP«. In: *Automata, Languages and Programming. 13th International Colloquium Rennes, France, July 15–-19, 1986 Proceedings*. Hrsg. von Laurent Kott. Bd. 58. Lecture Notes in Computer

Science 1. Berlin, Heidelberg: Springer, 1986, S. 123–135. DOI: https://doi.org/10.1016/0304-3975(88)90022-9.

[KMT03] Johannes Köbler, Jochen Messner und Jacobo Torán. »Optimal proof systems imply complete sets for promise classes«. In: *Information and Computation* 184.1 (2003), S. 71–92. DOI: 10.1016/S0890-5401(03)00058-0.

[KP89] Jan Krajíček und Pavel Pudlák. »Propositional proof systems, the consistency of first order theories and the complexity of computations«. In: *Journal of Symbolic Logic* 54.3 (1989), S. 1063–1079. DOI: 10.2307/2274765.

[Per14] Sylvain Perifel. *Complexité algorithmique*. Paris: Ellipses, 2014. ISBN: 978-2-7298-8692-9. URL: https://www.irif.fr/~sperifel/complexite.pdf.

[Pud17] Pavel Pudlák. »Incompleteness in the finite domain«. In: *The Bulletin of Symbolic Logic* 23.4 (2017), S. 405–441. DOI: 10.1017/bsl.2017.32.

[Raz94] Alexander A. Razborov. »On provably disjoint NP-pairs«. In: *BRICS Report Series* 1.36 (1994). DOI: 10.7146/brics.v1i36.21607.

[Sip82] Michael Sipser. »On relativization and the existence of complete sets«. In: *Automata, Languages and Programming*. Hrsg. von Mogens Nielsen und Erik Meineche Schmidt. Bd. 140. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1982, S. 523–531. DOI: 10.1007/BFb0012797.

[Val76] Leslie G. Valiant. »Relative complexity of checking and evaluating«. In: *Information Processing Letters* 5.1 (1976), S. 20–23. DOI: 10.1016/0020-0190(76)90097-1.

[Wec00] Gerd Wechsung. *Vorlesungen zur Komplexitätstheorie*. Bearb. von Johannes Buchmann u. a. Bd. 32. Wiesbaden: Vieweg+Teubner Verlag, 2000. ISBN: 978-3-519-00315-1. DOI: 10.1007/978-3-322-80024-4.

Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt zu haben.

Würzburg, den 02. Juli 2019