Albert-Ludwigs-Universität Freiburg

Mathematisches Institut

Komplexitätstheorie über beliebigen Strukturen

Diplomarbeit

von

Björn Lellmann

Betreuer: Prof. Dr. Jörg Flum
Abgabetermin: 16. April 2008

An dieser Stelle möchte ich zunächst Herrn Prof. Dr. Jörg Flum für die interessante Themenstellung und das gute Arbeitsumfeld danken. Desweiteren schulde ich Cordula Ritter und Moritz Müller vom Mathematischen Institut einen besonderen Dank für ihre Unterstützung. Insbesondere danke ich Moritz Müller für die Zusammenarbeit beim Beweis des Transfertheorems und für seine anregenden Denkanstöße. Nicht zuletzt danke ich all denen, die sich die Zeit genommen haben, die Arbeit korrekturzulesen.

Hiermit erkläre ich, dass ich die Diplomarbeit selbstständig angefertigt und nur die angegebenen Quellen verwendet habe.

Freiburg, den 16. April 2008

Björn Lellmann

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die klassische Komplexitätstheorie beschäftigt sich mit der Komplexität von Algorithmen, welche aus endlichen 0-1-Folgen neue endliche 0-1-Folgen produzieren. Da die Konzentration auf derartige Algorithmen manchmal limitierend ist, wurden zunächst von L. Blum, F. Cucker und S. Smale in ihrem Artikel [2] Algorithmen über beliebigen Ringen untersucht. Dieser Ansatz wurde später von J.B. Goode in [15] und von B. Poizat in [22] weiter dahingehend verallgemeinert, daß den betrachteten Algorithmen (fast) beliebige Strukturen zugrundeliegen konnten.

Die Formalisierung des informellen Algorithmusbegriffes wurde dabei einerseits von Blum, Shub und Smale bewerkstelligt, indem Flußdiagramme, welche ein Programm einer Maschine beschreiben, verallgemeinert wurden. Andererseits wurden unter anderem von Poizat das Konzept der klassischen Turingmaschine mit einem den Algorithmus beschreibenden Programm und ebenso das Konzept der den Algorithmus darstellenden Schaltkreise verallgemeinert. Allerdings ist es bei allen diesen Konzepten erlaubt, Elemente des Universums der zugrundeliegenden Struktur in die Berechnung mit einzubeziehen, welche nicht Teil der Eingabe oder Konstanten der Struktur sind. Ein derartiger Algorithmus ist nun äquivalent zu einem Algorithmus über der jeweiligen mit den entsprechenden Konstanten verlängerten Struktur, welcher in seiner Berechnung nur auf seine Eingabe und die Konstanten der Struktur zugreift. Es ist also auch interessant, solche „parameterfreien“ Algorithmen zu untersuchen. Ein derartiger Ansatz liegt zum Beispiel den Arbeiten [14] von P. Koiran oder [16] von A. Hemmerling zugrunde.

Wir werden im folgenden diesem Verständnis folgen und zunächst in Kapitel 2 den informellen Algorithmusbegriff in Anlehnung an [22] mittels des Begriffs der deterministischen Turingmaschine über einer beliebigen Struktur formal fassen. Weiter werden wir das klassische Konzept der nichtdeterministischen Turingmaschine auf beliebige Strukturen verallgemeinern. Mit diesem Rüstzeug können wir dann analog zu den klassischen Komplexitätsklassen P und NP die Klassen $\mathrm{P}_{\mathcal{M}}$ und $\mathrm{NP}_{\mathcal{M}}$ über einer beliebigen Struktur $\mathcal{M}$ untersuchen. In diesen wird versucht, die informellen Vorstellungen von schnell entscheidbaren Problemen und von mittels glücklichen Ratens schnell entscheidbaren Problemen zu formalisieren. Wir werden sehen, daß die Probleme, welche mittels glücklichen Ratens schnell entscheidbar sind, gerade diejenigen Probleme sind, deren Ja-Instanzen bei Vorlage eines geeigneten Zeugens schnell verifiziert werden können.

Anschließend werden wir in Kapitel 3 in Anlehnung an [2] und [3] den Algorithmusbe-

griff nicht durch Programmcode sondern durch Flußdiagramme formalisieren. Wir werden sehen, daß diese beiden Modelle bis auf einen polynomiellen Zeitverlust äquivalent sind. Das zu den nichtdeterministischen Turingmaschinen analoge Konzept werden wir auch von der Darstellung durch Flußdiagramme ausgehend einführen. Auch diese beiden Modelle werden sich als äquivalent erweisen.

Die großen noch offenen Probleme der klassischen Komplexitätstheorie sind die Frage, ob die beiden Klassen P und NP übereinstimmen, und die Verallgemeinerung hiervon, die Frage ob die polynomielle Hierarchie auf irgendeiner Ebene kollabiert. Die analogen Fragen lassen sich nun auch bezüglich einer beliebigen Struktur $\mathcal{M}$ stellen: Gilt $\mathrm{P}_{\mathcal{M}} = \mathrm{NP}_{\mathcal{M}}$, und kollabiert die polynomielle Hierarchie im Sinne von $\mathcal{M}$ auf irgendeine Ebene? In Kapitel 4 werden wir uns in großen Teilen der Darstellung in [22] folgend mit diesen Problemen beschäftigen. Wir werden mit den Standardtypen Kriterien an Strukturen kennenlernen, welche sicherstellen, daß diese Strukturen sich bezüglich eines Kollaps' der polynomiellen Hierarchie auf eine bestimmte Ebene genauso verhalten wie die Standardstruktur. Die Beantwortung dieser Frage in einer solchen Struktur ist also äquivalent zu der Beantwortung der entsprechenden Frage in der klassischen Komplexitätstheorie. Schließlich werden wir ein Transfertheorem für elementar äquivalente Strukturen zeigen, nach welchem die Antwort auf die Frage, ob die polynomielle Hierarchie auf eine bestimmte Ebene kollabiert, in allen elementar äquivalenten Strukturen dieselbe ist. Dieses Ergebnis wurde zunächst von A. Hemmerling in [16] mit Hilfe von Berechnungsbäumen gezeigt, unser Beweis verallgemeinert dagegen den Beweis des Transfertheorems für algebraisch abgeschlossene Körper der Charakteristik 0 aus [3].

Abschließend werden wir in Kapitel 5 verschiedene Strukturen in Hinblick darauf untersuchen, ob sie die in den Standardtypen festgelegten Kriterien erfüllen. Ein Hauptaugenmerk wird dabei auf Strukturen über den reellen Zahlen liegen. Dieses Kapitel stützt sich im wesentlichen auf [3] und [22].

1.1 Grundlagen

Zunächst wollen wir uns über die im folgenden verwendeten Grundlagen verständigen. Mit $\mathbb{N}$ bezeichnen wir die Menge der natürlichen Zahlen $\{0, 1, 2, \ldots\}$, mit $\mathbb{N}_{>0}$ die positiven natürlichen Zahlen, die natürlichen Zahlen ohne die 0. Als abkürzende Schreibweise verwenden wir das Symbol $[k]$ für die Menge $\{1, 2, \ldots, k\}$, wenn k eine natürliche Zahl ist. Für eine reelle Zahl r ist $\lceil r \rceil := \min\{z \in \mathbb{Z} \mid r \leq z\}$, und $\lfloor r \rfloor := \max\{z \in \mathbb{Z} \mid z \leq r\}$.

Weiterhin gehen wir von einem grundlegenden Verständnis der Logik der ersten Stufe aus, wie es zum Beispiel in [11] vermittelt wird. Strukturen stellen wir dar als $\mathcal{M}, \mathcal{N}, \ldots$. Hierbei ist $\mathcal{M} = (M, (f_i^{\mathcal{M}})_{i \in I_1}, (R_i^{\mathcal{M}})_{i \in I_2}, (c_i^{\mathcal{M}})_{i \in I_3})$, wobei die nichtleere Menge M das Universum von $\mathcal{M}$ bezeichnet, die Funktionen $(f_i^{\mathcal{M}})_{i \in I_1}$ die Interpretationen der Funktionssymbole $(f_i)_{i \in I_1}$ aus der Symbolmenge von $\mathcal{M}$, die Relationen $(R_i^{\mathcal{M}})_{i \in I_2}$ analog die Interpretationen der Relationssymbole $(R_i)_{i \in I_2}$ und die Elemente $(c_i^{\mathcal{M}})_{i \in I_3}$ des Universums von $\mathcal{M}$ die Interpretationen der Konstantensymbole $(c_i)_{i \in I_3}$ der Symbolmenge von $\mathcal{M}$. Da normalerweise aus dem Kontext hervorgehen wird, ob es sich um Symbole oder ihre Interpretationen handelt, lassen wir in der Regel die Kennzeichnung $^{\mathcal{M}}$ der Interpretationen weg. Der Übersichtlichkeit wegen fassen wir Konstanten oft als nullstellige Funktionen

auf. Mit $(\mathcal{M}, (c_m)_{m\in M})$ bezeichnen wir die mit den Elementen ihres Universums als Konstanten verlängerte Struktur. Falls für zwei Strukturen $\mathcal{M}$ und $\mathcal{N}$ derselben Symbolmenge gilt, daß das Universum von $\mathcal{M}$ im Universum von $\mathcal{N}$ enthalten ist und die Interpretationen der Funktions-, Relations- und Konstantensymbole sich auf dem Universum von $\mathcal{M}$ gleich verhalten, so nennen wir $\mathcal{M}$ eine Substruktur von $\mathcal{N}$ und schreiben $\mathcal{M} \subseteq \mathcal{N}$. Die Struktur $\mathcal{N}$ nennen wir dann auch Oberstruktur von $\mathcal{M}$. Für eine Struktur $\mathcal{M}$ und eine Teilmenge X des Universums von $\mathcal{M}$ bezeichnen wir mit $[X]^{\mathcal{M}}$ die von der Menge X in $\mathcal{M}$ erzeugte Substruktur.

Erststufige Formeln der Sprache einer Struktur $\mathcal{M}$ werden wie gewohnt definiert und von uns mit $\varphi, \psi, \ldots$ bezeichnet. Die Menge der freien Variablen einer Formel φ ist dann die Menge derjenigen Variablen, für die mindestens ein Vorkommnis in φ nicht durch einen Existenzquantor oder einen Allquantor quantifiziert ist. Hat eine Formel φ die freien Variablen $x_1, \ldots, x_n$, so schreiben wir auch $\varphi(x_1, \ldots, x_n)$. Ist $\varphi(x_1, \ldots, x_n)$ eine Formel und sind $a_1, \ldots, a_n$ Elemente aus M, so schreiben wir $\mathcal{M} \models \varphi(\bar{a})$, falls die durch Substitution jedes nicht quantifizierten Vorkommens von $x_1, \ldots, x_n$ in φ durch $a_1, \ldots, a_n$ gewonnene Aussage in der Struktur $\mathcal{M}$ gilt. Für eine Konjunktion oder Disjunktion von endlich vielen Formeln $\varphi_1, \ldots, \varphi_n$ schreiben wir auch $\bigwedge_{i\in[n]} \varphi_i$ beziehungsweise $\bigvee_{i\in[n]} \varphi_i$.

Wir werden Berechnungen über beliebigen Strukturen $\mathcal{M}$ mit Universum M und Symbolmenge Σ betrachten, wobei zwei Konstantensymbole „0“ und „1“, ein Symbol „=“ für die Gleichheitsrelation, ein einstelliges Funktionssymbol „Id“ für die Identitätsfunktion, welche jedes Element auf sich selbst abbildet, und ein dreistelliges Funktionssymbol „S“ für die Selektorfunktion

$$S(x, y, z) = \begin{cases} y, & \text{falls } x = 0 \\ z, & \text{falls } x = 1 \\ x, & \text{sonst} \end{cases}$$

in Σ enthalten sind. Außerdem enthält Σ nur endlich viele mehr als nullstellige Funktions- und Relationssymbole. Die Symbolmenge kann jedoch beliebig viele nullstellige Funktionssymbole beziehungsweise Konstantensymbole enthalten. Da die Symbole $0, 1, \mathrm{Id}, S, =$ in jeder der betrachteten Symbolmengen enthalten sind, lassen wir sie bei der Angabe einer Struktur normalerweise weg. Die Struktur $(\{0,1\}, \neg, \vee, \wedge)$ mit Universum $\{0,1\}$ und den üblichen booleschen Funktionen bezeichnen wir als Standardstruktur.

Bemerkung 1.1.1 In Strukturen mit der Selektorfunktion S und den Konstanten 0 und 1 sind insbesondere die auf $\{0,1\}$ definierten booleschen Funktionen $\neg, \vee, \wedge$ interpretierbar durch:

$$\begin{aligned} \neg x &= S(x, 1, 0) \\ x \vee y &= S(x, y, 1) \\ x \wedge y &= S(x, 0, y) \end{aligned}$$

Da wir davon ausgehen, daß die Selektorfunktion und die Konstanten 0 und 1 in jeder der betrachteten Strukturen zur Verfügung steht, können wir somit in jeder der betrachteten Strukturen Berechnungen im Sinne der Standardstruktur ausführen.

Das Universum M einer Struktur $\mathcal{M}$ wird auch als *Alphabet* betrachtet, die Menge $M^* := \{\lambda\} \cup \bigcup_{n\in\mathbb{N}} \{a_0 \ldots a_n \mid a_i \in M \text{ für } i \in \{0\} \cup [n]\}$ enthält alle endlichen Ketten von

Zeichen aus M und das Symbol λ für die Zeichenkette der Länge null. Solche endliche Zeichenketten werden auch als *Wörter* bezeichnet, mit $|\bar{a}|$ bezeichnen wir die Länge des Wortes $\bar{a}$. Das mit λ bezeichnete Wort der Länge null nennen wir auch das *leere Wort*. Eine *Sprache* $L \subseteq M^*$ ist dann eine Menge von Wörtern. Wir bezeichnen Sprachen auch als *Probleme*. Wörter aus $\{0,1\}^*$ bezeichnen wir auch als boolesche Wörter. Wenn in bezug auf eine Struktur $\mathcal{M}$ von einer charakteristischen Funktion einer Menge $X \subseteq M^*$ die Rede ist, so ist immer die Funktion $\chi_X : M^* \to \{0,1\}$ mit

$$\chi_X(x) = \begin{cases} 1, & \text{falls } x \in X \\ 0, & \text{sonst} \end{cases}$$

gemeint.

Da wir manchmal nicht an einzelnen Funktionen, sondern an der Größenordnung bestimmter Funktionen, an der Geschwindigkeit, mit der sie wachsen, interessiert sind, ist es nützlich, die $\mathcal{O}$-Notation einzuführen. Für zwei Funktionen $f : \mathbb{N} \to \mathbb{N}$ und $g : \mathbb{N} \to \mathbb{N}$ schreiben wir also $f(n) = \mathcal{O}(g(n))$, falls es natürliche Zahlen c und n_0 gibt, so daß für alle natürlichen Zahlen $n > n_0$ gilt: $f(n) \leq c \cdot g(n)$. Mit dem Logarithmus schließlich ist immer der Logarithmus zur Basis 2 gemeint, und wir schreiben $\log(x)$ für $\lceil \log(x) \rceil$.

1.1.1 Schaltkreise

Wir werden später das Konzept der Schaltkreise über einer Struktur $\mathcal{M}$ verwenden. Da das Modell der formalen Fassung eines Algorithmus mittels eines Schaltkreises jedoch nicht im Zentrum unserer Aufmerksamkeit steht, führen wir an dieser Stelle nur kurz die Grundlagen ein. Für eine ausführlichere Darstellung siehe zum Beispiel [22]. An einem einen Algorithmus darstellenden Schaltkreis kann man auf relativ übersichtliche Weise gewisse Eigenschaften des entsprechenden Algorithmus ablesen, wie zum Beispiel die benötigte sequentielle Zeit oder die benötigte Parallelzeit. Allerdings hat das Schaltkreismodell den Nachteil, daß ein Schaltkreis nur mit Eingaben einer festen Länge rechnen kann, das Berechnungsmodell ist nicht uniform.

Schaltkreise werden über gerichtete Graphen definiert, deswegen als kurze Erinnerung: Ein *gerichteter Graph* ist eine Menge V von Knoten mit einer zweistelligen Relation E. Diese Relation gibt an, welche Knoten mit einer gerichteten Kante verbunden sind. Gerichtete Kanten nennen wir im folgenden auch *Pfeile*. Ein *gerichteter Zykel* ist dann eine Menge $\{a_1, \ldots, a_k\}$ von Knoten, mit $(a_1, a_2) \in E, (a_2, a_3) \in E, \ldots, (a_{k-1}, a_k) \in E, (a_k, a_1) \in E$.

Definition 1.1.2 (Schaltkreis im Sinne von $\mathcal{M}$) Ein *Schaltkreis im Sinne der Struktur* $\mathcal{M}$ ist ein gerichteter Graph ohne gerichtete Zykel, dessen Knoten in einer der folgenden Weisen gelabelt sind:

(i) Knoten, an denen keine Pfeile enden, heißen *Eingangsknoten* (oder *Inputknoten*), und sind mit jeweils einer Variable oder einer nullstelligen Funktion (einer Konstanten) aus $\mathcal{M}$ gelabelt

(ii) Knoten, an denen k Pfeile enden, sind mit einer k-stelligen Funktion aus $\mathcal{M}$ oder einer charakteristischen Funktion einer k-stelligen Relationen aus $\mathcal{M}$ gelabelt.

Da im Allgemeinen die Reihenfolge der Inputs für das Ergebnis einer Funktion wichtig ist, gehen wir davon aus, daß es unter den an einem Knoten endenden Pfeilen eine Reihenfolge gibt. Knoten, von denen keine Pfeile ausgehen, heißen *Ausgangsknoten* und geben das Ergebnis der Berechnung des Schaltkreises an. Die *Größe* eines Schaltkreises C (also die Anzahl seiner Knoten) bezeichnen wir mit $t(C)$. Schaltkreise, die keine mit Konstantensymbolen gelabelten Inputknoten haben, heißen *konstantenfrei*.

Ein Schaltkreis C mit k Inputknoten und l Outputknoten berechnet eine Funktion $f_C : M^k \mapsto M^l$. Man kann einen Schaltkreis mit einem Outputknoten als eine bestimmte Menge $X \subseteq M^n$ charakterisierend auffassen:

Definition 1.1.3 Ein Schaltkreis $C(x_1, \ldots, x_n)$ mit einem Outputknoten *akzeptiert eine Menge* $X \subseteq M^n$, falls für alle $\bar{x} \in M^n$ gilt: $C(\bar{x}) = 1 \Leftrightarrow \bar{x} \in X$.

Wir werden unter anderem in Abschnitt 4.4 eine einfache Eigenschaft von solchen Schaltkreisen benötigen, die wir an dieser Stelle jedoch nicht beweisen:

Bemerkung 1.1.4 Zu jedem Schaltkreis C über einer Struktur $\mathcal{M}$ mit n Inputknoten und einem Outputknoten läßt sich eine quantorenfreie Formel ψ_C mit n freien Variablen konstruieren, so daß für alle $\bar{x} \in M^n$ gilt:

$$C(\bar{x}) = 1 \Leftrightarrow \mathcal{M} \models \psi_C(\bar{x}).$$

Diese Formel kann allerdings sehr lang sein. Hierfür siehe auch [22], S.91 oder [26], Lemma 26.

Da die Anzahl der Inputknoten eines Schaltkreises fest ist, kann man mit einzelnen Schaltkreisen leider nur Sprachen, deren Elemente alle die selbe Länge haben, charakterisieren. Um allgemein Sprachen, die Wörter unterschiedlicher Länge beinhalten, zu charakterisieren, benötigt man deshalb eine Familie von Schaltkreisen.

Definition 1.1.5 (Schaltkreisfamilie) Eine *Schaltkreisfamilie über* $\mathcal{M}$ ist eine Folge $(C_n)_{n\in\mathbb{N}_{>0}}$ von Schaltkreisen über $\mathcal{M}$, wobei für alle $n \in \mathbb{N}_{>0}$ der Schaltkreis C_n genau n Inputknoten besitzt. Eine Schaltkreisfamilie $(C_n)_{n\in\mathbb{N}_{>0}}$ *akzeptiert eine Sprache* $L \subseteq M^*$, falls für jedes $n \in \mathbb{N}_{>0}$ und für jedes $\bar{x} \in M^n$ gilt: $C_n(\bar{x}) = 1 \Leftrightarrow \bar{x} \in L$.

Oft ist es nützlich, wenn man den Begriff einer Schaltkreisfamilie dahingehend einschränkt, daß jeder der Schaltkreise nur bestimmte Konstanten verwenden darf:

Definition 1.1.6 (Gesetzte Schaltkreisfamilie) Eine *gesetzte Schaltkreisfamilie über* $\mathcal{M}$ ist eine Schaltkreisfamilie $(C_n)_{n\in\mathbb{N}_{>0}}$ über $\mathcal{M}$, für die es eine endliche Menge K von Konstantensymbolen aus der Symbolmenge von $\mathcal{M}$ gibt, so daß die Inputknoten eines jeden Schaltkreises aus $(C_n)_{n\in\mathbb{N}_{>0}}$ mit Variablen oder Konstantensymbolen aus K gelabelt sind.

Eine gesetzte Schaltkreisfamilie lässt sich beschreiben als eine Familie $(C'_n(\bar{x}, \bar{y}))_{n\in\mathbb{N}_{>0}}$ von konstantenfreien Schaltkreisen über $\mathcal{M}$ zusammen mit $|\bar{x}|$-vielen Konstanten $c_1, \ldots, c_{|\bar{x}|}$ aus $\mathcal{M}$. Über einer Struktur, deren Symbolmenge nur endlich viele Konstantensymbole enthält, ist jede Schaltkreisfamilie gesetzt.

Wenn es in einer Struktur überabzählbar viele Konstanten gibt, so kann eine Schaltkreisfamilie nicht mehr binär kodiert werden. Eine gesetzte Schaltkreisfamilie dagegen kann immer binär kodiert werden, wenn die Menge K bekannt ist.

1.1.2 Nicht-uniforme Komplexitätsklassen

Mit Hilfe des Begriffs des Schaltkreises läßt sich nun der Begriff der Berechenbarkeit über einer Struktur $\mathcal{M}$ formal fassen: eine Funktion $f : M^* \to M^*$ ist berechenbar, wenn es eine Familie von Schaltkreisen $(C_n)_{n\in\mathbb{N}_{>0}}$ im Sinne von $\mathcal{M}$ gibt, so daß für alle $\bar{x} \in M^*$ die Gleichheit $f(\bar{x}) = C_{|\bar{x}|}(\bar{x})$ gilt. Weiter lassen sich nicht-uniforme Komplexitätsklassen definieren. Die Fragestellung ist hierbei, für welche Sprachen es Familien von „handlichen" Schaltkreisen gibt, die die jeweilige Sprache entscheiden. Ein „handlicher Schaltkreis" ist dabei ein Schaltkreis, der verglichen mit der Anzahl seiner Inputknoten nicht zu groß ist. Für eine Familie von Schaltkreisen bedeutet dies, daß die Größe der Schaltkreise höchstens polynomiell in der Anzahl der Inputknoten wächst.

Definition 1.1.7 ($\mathbb{P}^*_{\mathcal{M}}$) Ein Problem $X \subseteq M^*$ ist in der Klasse $\mathbb{P}^*_{\mathcal{M}}$, falls es ein Polynom p und eine Familie $(C_n)_{n\in\mathbb{N}_{>0}}$ von Schaltkreisen mit $t(C_n) \leq p(n)$ gibt, welche das Problem X akzeptiert.

Im Falle von gesetzten Schaltkreisen:

Definition 1.1.8 ($\mathbb{P}_{\mathcal{M}}$) Ein Problem $X \subseteq M^*$ ist in der Klasse $\mathbb{P}_{\mathcal{M}}$, falls es ein Polynom p und eine gesetzte Familie $(C_n)_{n\in\mathbb{N}_{>0}}$ von Schaltkreisen mit $t(C_n) \leq p(n)$ gibt, welche das Problem X akzeptiert.

Da eine Familie von gesetzten Schaltkreisen sicher auch eine Familie von Schaltkreisen ist, gilt offensichtlicherweise $\mathbb{P}_{\mathcal{M}} \subseteq \mathbb{P}^*_{\mathcal{M}}$.

Bemerkung 1.1.9 ($|\mathbb{P}_{\mathcal{M}}| \geq 2^{\aleph_0}$) In jeder Struktur sind Probleme, deren Lösung nur von der Länge der Eingabe abhängt, in der Klasse $\mathbb{P}_{\mathcal{M}}$. Ein Schaltkreis mit n Eingabeknoten gibt je nachdem, ob Eingaben der Länge n Ja-Instanzen des Problems sind, einfach immer 0 beziehungsweise 1 aus. Somit liegen allerdings in jeder beliebigen Struktur mindestens $2^{\aleph_0}$ viele Probleme in der Klasse $\mathbb{P}_{\mathcal{M}}$.

Während die Klasse $\mathbb{P}_{\mathcal{M}}$ das nicht uniforme Analogon zu der klassischen Komplexitätsklasse P ist, ist die Klasse $\mathbb{NP}_{\mathcal{M}}$ das nicht uniforme Gegenstück zu der klassischen Komplexitätsklasse NP. Sie wird über die Existenz eines Zeugens für Ja-Instanzen definiert:

Definition 1.1.10 ($\mathbb{NP}_{\mathcal{M}}$) Ein Problem $X \subseteq M^*$ ist in der Klasse $\mathbb{NP}_{\mathcal{M}}$, falls es ein Problem $Y \in \mathbb{P}_{\mathcal{M}}$ und ein Polynom q gibt, so daß für alle $\bar{x} \in M^*$ gilt: $\bar{x} \in X \Leftrightarrow$ es gibt ein $\bar{y} \in M^{q(|\bar{x}|)}$ mit $\bar{x}\bar{y} \in Y$.

Manchmal werden wir auch verlangen, daß es für Ja-Instanzen einen booleschen Zeugen gibt. Dies liefert die Klasse $\mathbb{BNP}_{\mathcal{M}}$:

Definition 1.1.11 ($\mathbb{BNP}_{\mathcal{M}}$) Ein Problem $X \subseteq M^*$ ist in der Klasse $\mathbb{BNP}_{\mathcal{M}}$, falls es ein Problem $Y \in \mathbb{P}_{\mathcal{M}}$ und ein Polynom q gibt, so daß für alle $\bar{x} \in M^*$ gilt: $\bar{x} \in X \Leftrightarrow$ es gibt ein $\bar{\varepsilon} \in \{0,1\}^{q(|\bar{x}|)}$ mit $\bar{x}\bar{\varepsilon} \in Y$.

Die Klassen $\mathbb{NP}^*_{\mathcal{M}}$ und $\mathbb{BNP}^*_{\mathcal{M}}$ werden analog definiert, allerdings müssen die Schaltkreisfamilien hierbei nicht gesetzt sein.

Kapitel 2

Turingmaschinen über $\mathcal{M}$

Wir werden nun zunächst das in Anlehnung an [22] definierte Konzept einer deterministischen Turingmaschine über einer Struktur $\mathcal{M}$ kennenlernen. Im Gegensatz zu der dort verwendeten Definition wird es einer Turingmaschine hier jedoch nicht erlaubt sein, auf Elemente der Struktur zurückzugreifen, die keine Konstanten der Struktur sind oder vorher aus der Eingabe und den Konstanten der Struktur berechnet wurden. Da wir nur Strukturen, welche die Konstanten 0 und 1 enthalten, betrachten, ist also jeder der hier betrachteten Turingmaschinen erlaubt, die Konstanten 0 und 1 in ihrer Berechnung zu verwenden. Die so definierten deterministischen Turingmaschinen werden dann (wiederum analog zu [22]) in Beziehung zu dem im vorherigen Abschnitt erwähnten Schaltkreismodell gesetzt und verwendet, um die grundlegenden Komplexitätsklassen $\mathrm{P}_{\mathcal{M}}$ und $\mathrm{EXP}_{\mathcal{M}}$ zu definieren.

Daraufhin werden wir zunächst binär nichtdeterministische Turingmaschinen über einer Struktur $\mathcal{M}$ und mit ihrer Hilfe die Komplexitätsklasse $\mathrm{BNP}_{\mathcal{M}}$ betrachten. Abschließend definieren wir vollständig nichtdeterministische Turingmaschinen über $\mathcal{M}$ und die entsprechende Komplexitätsklasse $\mathrm{NP}_{\mathcal{M}}$. Dabei werden wir jeweils sehen, daß die Klassen $\mathrm{BNP}_{\mathcal{M}}$ und $\mathrm{NP}_{\mathcal{M}}$ den Klassen $\mathrm{BVP}_{\mathcal{M}}$ beziehungsweise $\mathrm{VP}_{\mathcal{M}}$ entsprechen, welche im Sinne des in [3] oder [22] benutzten Begriffs von binärem und vollständigem Nichtdeterminismus definiert werden.

2.1 Deterministische Turingmaschinen

Definition 2.1.1 (Deterministische Turingmaschinen über $\mathcal{M}$) Eine *deterministische Turingmaschine über $\mathcal{M}$* besteht aus endlich vielen und mindestens zwei Bändern, welche jeweils in Felder unterteilt und beidseitig unbeschränkt sind. Jedes Feld kann leer sein oder ein Element der Struktur enthalten. Für jedes Band gibt es weiterhin einen Zeiger, welcher auf ein Feld des Bandes zeigt. Die Maschine kann verschiedene Aktionen ausführen. Welche Aktionen durchgeführt werden, wird in einem zugehörigen *Programm* festgelegt, welches Schritt für Schritt durchlaufen wird.

Etwas formaler wird jedes der k Bänder der Maschine als eine Kopie von $\mathbb{Z}$ aufgefasst. Die Funktionen $b_i : \mathbb{Z} \mapsto M \cup \{\lambda\}$ ordnen für $i \in [k]$ jeweils dem j-ten Feld des i-ten Bandes dessen Inhalt $b_i(j)$ zu, wobei λ für „leer“ steht. Im folgenden sind mit den *vom Zeiger markierten n Feldern* eines Bandes i die n Felder beginnend mit und rechts des Zeigers

des i-ten Bandes gemeint. Zeigt der Zeiger dieses Bandes also zum Beispiel auf das Feld m, so sind die von diesem Zeiger markierten n Felder die Felder $m, m+1, \ldots, m+(n-1)$ des i-ten Bandes.

Das zugehörige Programm ist eine endliche Folge $(1, \Pi_1), \ldots, (Q, \Pi_Q)$ von Anweisungen, wobei $1, \ldots, Q$ die Zeilennummern und $\Pi_1, \ldots, \Pi_Q$ die entsprechenden Befehle sind. Ohne Einschränkung ist dabei $\Pi_Q =$`(stop)`. Ein Befehl Π_l hat dabei eine der folgenden Formen und Bedeutungen:

- `(move` i `right)` oder `(move` i `left)`: bewegt den Zeiger des i-ten Bandes um eine Position nach rechts beziehungsweise nach links und fährt mit Programmzeile $l+1$ fort.

- `(compute` f`,`i `to` j`)` für eine n-stellige Funktion f der Struktur und $i, j \in [k]$: berechnet den Wert der Funktion f angewendet auf die Inhalte der durch den Zeiger des i-ten Bandes markierten n Felder, und schreibt diesen Wert in das durch den Zeiger des j-ten Bandes markierte Feld. Sollte eines oder mehrere der durch den Zeiger des i-ten Bandes markierten n Felder leer sein, so wird stattdessen an der entsprechenden Stelle der Funktion als Argument „0“ eingesetzt, das entsprechende Feld bleibt leer. Da wir Konstanten als nullstellige Funktionen auffassen, kann mit diesem Befehl auch eine Konstante auf das entsprechende Feld geschrieben werden. Die nächste Programmzeile ist die Zeile $l+1$.

- `(if` R`,`i `then` p`, else` q`)` für eine n-stellige Relation R der Struktur und $p, q \in [Q]$: testet, ob die Relation R auf die Inhalte der durch den Zeiger des i-ten Bandes markierten n Felder zutrifft. Ist dies der Fall, so springt das Programm in die p-te Zeile, andernfalls in die q-te Zeile.

- `(if` $i = \lambda$ `then` p`, else` q`)` für $i \in [k]$ und $p, q \in [Q]$: testet, ob das Feld, auf welches der Zeiger des i-ten Bandes zeigt, leer ist. Ist dies der Fall, so springt das Programm in die p-te Zeile, andernfalls in die q-te Zeile.

- `(stop)`: hält die Maschine an. Aus technischen Gründen setzen wir als nächste Programmzeile die Zeile l.

Eine *Konfiguration* der Maschine ist eine Beschreibung des Zustandes des Maschine zu einem bestimmten Zeitpunkt der Berechnung, bestehend aus der momentanen Zeilennummer, der momentanen Position der Zeiger aller Bänder und den Inhalten der Felder aller Bänder. Formaler ist eine Konfiguration ein Tupel $(q, j_1, \ldots, j_k, b_1, \ldots, b_k)$ mit $q \in [Q]$ für die Zeilennummer, $j_1, \ldots, j_k \in \mathbb{Z}$ für die Positionen der Zeiger und $b_i : \mathbb{Z} \mapsto M \cup \{\lambda\}$, $b_i^{-1}(M)$ endlich für $i \in [k]$ für die Inhalte der Felder der jeweiligen Bänder.

Mittels des Begriffs der Konfiguration lässt sich eine Berechnung der Maschine folgendermaßen formal beschreiben: eine *Startkonfiguration* beschreibt den Zustand zu Beginn der Berechnung, ist also eine Konfiguration mit Zeilennummer $q = 1$. Zu Beginn stehen die Zeiger aller Bänder standardmäßig auf der Position 0, und alle Felder aller Bänder sind leer bis auf die Felder $0, \ldots, n$ des ersten Bandes, welche Elemente $a_1, \ldots, a_n$ aus M enthalten. Das Wort $\bar{a} = a_1 \ldots a_n$ nennen wir die *Eingabe* der Maschine. In einer Startkonfiguration ist also $j_1 = j_2 = \ldots = j_k = 0$ und $b_1(1) = a_1, \ldots, b_1(n) = a_n, b_1(j) =$

λ für $j \notin \{1, \ldots, n\}$ und $b_i \equiv \lambda$ für $i > 1$, wobei $a_1 \ldots a_n$ die Eingabe ist. Eine *Nachfolgekonfiguration* einer Konfiguration K ist eine Konfiguration, welche durch Anwenden des zur Zeilennummer in K gehörigen Programmbefehls auf die durch K festgelegten Zeigerpositionen und Inhalte der Felder hervorgehen kann. Bei einer deterministischen Turingmaschine ist die jeweils nächste Konfiguration bei gegebener Eingabe durch den entsprechenden Befehl des Programmes eindeutig bestimmt. Es gibt also eine (von der konkreten Maschine abhängige) *Übergangsfunktion*, welche auf der Menge der möglichen Konfigurationen der Maschine definiert ist und jeder dieser Konfigurationen ihre Nachfolgekonfiguration zuordnet. Als *Haltekonfiguration* schließlich bezeichnen wir eine Konfiguration, bei der der durch die in der Konfiguration bestimmte Zeilennummer geforderte Befehl der Befehl `(stop)` ist.

Eine *Berechnung* der Maschine ist dann eine endliche Folge $K_0, \ldots, K_m$ von Konfigurationen, wobei K_0 eine Startkonfiguration mit der entsprechenden Eingabe und K_m eine Haltekonfiguration sind, und für alle $i \in [m]$ die Konfiguration K_i eine Nachfolgekonfiguration von K_{i-1} ist. Die *Ausgabe* besteht in diesem Fall aus den Inhalten der nichtleeren Felder beginnend mit und rechts der Position des Zeigers des k-ten Bandes, oder formaler aus dem Wort $b_k(j_k)b_k(j_k+1)\ldots b_k(\min\{z > j_k \mid b_k(z+1) = \lambda\})$. Wir sagen dann, daß die Maschine das entsprechende Wort *ausgibt*. Wir wollen weiterhin sagen, daß eine deterministische Turingmaschine angesetzt auf eine Eingabe $\bar{a} \in M^*$ *schließlich hält*, wenn es eine Berechnung der Maschine mit der Startkonfiguration mit der Eingabe $\bar{a}$ gibt, das heißt wenn die Maschine angesetzt auf die Eingabe $\bar{a}$ nach endlich vielen Schritten eine Haltekonfiguration erreicht. Entsprechend sagen wir $\mathbb{A}$ *hält bei Eingabe von* $\bar{a} \in M^*$ *nach* t *Schritten*, wenn es eine aus t Konfigurationen bestehende Berechnung von $\mathbb{A}$ gibt, so daß die Startkonfiguration die Eingabe $\bar{a}$ bestimmt. Wenn $\mathbb{A}$ bei Eingabe von $\bar{a}$ nach t Schritten hält und $\bar{b}$ ausgibt, schreiben wir auch $\mathbb{A}(\bar{a}) = \bar{b}$ *nach* t *Schritten*. Da bei einer gegebenen Eingabe alle Konfigurationen, die von der Maschine angesetzt auf die entsprechende Eingabe durchlaufen werden, eindeutig bestimmt sind, sind für jede Eingabe alle diese Konfigurationen eindeutig. Insbesondere ist für jede Eingabe, für die die Maschine angesetzt auf ebendiese schließlich hält, die Ausgabe der Maschine eindeutig.

Bemerkung 2.1.2 Betrachtet man als Struktur $\mathcal{M}$ die Standardstruktur $(\{0,1\}, \neg, \vee, \wedge)$, so erhält man mit unserer Definition eine deterministische Turingmaschine über $\mathcal{M}$, die dasselbe leistet, wie eine deterministische Turingmaschine im Sinne der klassischen Komplexitätstheorie. Für die klassische Definition siehe zum Beispiel [21].

Da bei einer deterministischen Turingmaschine über $\mathcal{M}$ für jede Eingabe, bei der die Maschine angesetzt auf diese schließlich hält, die Berechnung der Maschine angesetzt auf diese eindeutig ist, lässt sich eine deterministische Turingmaschine als Beschreibung einer partiellen Funktion von $U \subseteq M^*$ nach M^* auffassen. Für den Fall, daß die Maschine angesetzt auf jede beliebige Eingabe aus M^* schließlich hält, ergibt sich:

Definition 2.1.3 Eine deterministische Turingmaschine $\mathbb{A}$ über der Struktur $\mathcal{M}$ *berechnet die Funktion* $f : M^* \mapsto M^*$, falls $\mathbb{A}$ angesetzt auf ein beliebiges Element x aus M^* schließlich hält und $f(x)$ ausgibt. Eine Funktion $f : M^* \mapsto M^*$ heißt *berechenbar*, falls es eine deterministische Turingmaschine über $\mathcal{M}$ gibt, die f berechnet. Die Maschine $\mathbb{A}$ *entscheidet ein Problem* $X \subseteq M^*$, falls sie die charakteristische Funktion χ_X von X

berechnet. Dementsprechend heißt ein Problem $X \subseteq M^*$ *entscheidbar*, falls es eine deterministische Turingmaschine über $\mathcal{M}$ gibt, die X entscheidet.

Betrachten wir zur Veranschaulichung des ganzen Konzeptes ein einfaches Beispiel:

Beispiel 1 Sei $\mathcal{M} = (\mathbb{R}, +)$. Nach unserer Konvention lassen wir die Konstantensymbole $0, 1$ und die Funktions- beziehungsweise Relationssymbole $\mathrm{Id}, S, =$ in der Angabe der Symbolmenge von $\mathcal{M}$ weg. Wir betrachten die Funktion $f : \mathbb{R}^* \mapsto \mathbb{R}^*$, die jedem Wort $x_1 \ldots x_n$ aus $\mathbb{R}^*$ die reelle Zahl $x_1 + x_2 + \ldots + x_n$ zuordnet. Eine diese Funktion berechnende deterministische Turingmaschine hat dann zum Beispiel vier Bänder, das erste Band wird nach Konvention als Eingabeband, das vierte Band als Ausgabeband benutzt. Die beiden anderen Bänder werden für die Berechnung benutzt. Die Maschine addiert nun sukzessive die Elemente der Eingabe, hält danach an und gibt das Ergebnis aus.

Genauer kopiert die Maschine zunächst mit den Anweisungen

(1,`(compute Id, 1 to 2)`)
(2,`(move 1 right)`)
(3,`(move 2 right)`)

das erste Element der Eingabe auf das zweite Band, und geht dann mit den Anweisungen

(4,`(if` $1 = \lambda$ `then 11, else 5)`)
(5,`(compute Id, 1 to 2)`)
(6,`(move 2 left)`)
(7,`(compute +, 2 to 3)`)
(8,`(compute Id, 3 to 2)`)
(9,`(move 2 right)`)
(10,`(move 1 right)`)
(11,`(compute Id, 2 to 4)`)
(12,`(stop)`)

in eine Schleife, welche prüft, ob die Eingabe schon vollständig abgearbeitet wurde, falls dies der Fall ist das berechnete Ergebnis ausgibt und anhält, andernfalls das nächste Element der Eingabe zum bisherigen Ergebnis hinzuaddiert.

Wie man an diesem Beispiel schon erahnen kann, kann die genaue Beschreibung einer deterministischen Turingmaschine mit Hilfe ihres Programms eine relativ aufwendige und manchmal undurchschaubare Angelegenheit werden. Aus diesem Grund werden die Maschinen im folgenden oft durch den entsprechenden informellen Algorithmus beschrieben.

Zur Beurteilung der Komplexität einer Berechnung benutzen wir die Anzahl der bei dieser Berechnung von der Maschine durchlaufenen Schritte:

Definition 2.1.4 Sei T eine Funktion von $\mathbb{N}$ nach $\mathbb{R}$. Eine deterministische Turingmaschine $\mathbb{A}$ über einer Struktur $\mathcal{M}$ berechnet eine Funktion $f : M^* \to M^*$ *in* $T(n)$ *Schritten*, falls $\mathbb{A}$ bei Eingabe eines Elementes $\bar{x} \in M^*$ nach höchstens $\lceil T(|\bar{x}|) \rceil$ Schritten hält und $f(\bar{x})$ ausgibt. Analog entscheidet $\mathbb{A}$ ein Problem $X \subseteq M^*$ in $T(n)$ Schritten, falls $\mathbb{A}$ die charakteristische Funktion χ_X von X in $T(n)$ Schritten berechnet.

Wichtige Klassen von Funktionen werden „schnell“ berechenbare Funktionen und aufwendigere Funktionen sein:

Definition 2.1.5 Eine Funktion $f : M^* \to M^*$ ist *in polynomieller Zeit im Sinne von $\mathcal{M}$ berechenbar*, falls es ein Polynom $p(n)$ und eine deterministische Turingmaschine $\mathbb{A}$ über $\mathcal{M}$ gibt, so daß die Maschine $\mathbb{A}$ die Funktion f in Zeit $p(n)$ berechnet. Eine Funktion $f : M^* \to M^*$ ist *in exponentieller Zeit im Sinne von $\mathcal{M}$ berechenbar*, falls es ein Polynom $p(n)$, eine Konstante $C \in \mathbb{R}$ und eine deterministische Turingmaschine $\mathbb{A}$ über $\mathcal{M}$ gibt, so daß die Maschine $\mathbb{A}$ die Funktion f in Zeit $C \cdot \exp(p(n))$ berechnet.

2.1.1 Der Zusammenhang zwischen Turingmaschinen- und Schaltkreiskonzept

Nachdem mit dem Schaltkreismodell und dem Turingmaschinenmodell zwei Modelle zur Formalisierung von Algorithmen eingeführt worden sind, betrachten wir nun den Zusammenhang zwischen diesen beiden Modellen.

Zunächst lässt sich ein konstantenfreier Schaltkreis über $\mathcal{M}$ binär kodieren. Eine Möglichkeit, dies zu tun, wäre, eine Liste der Knoten des Schaltkreises mit ihrem jeweiligen Label und den Knoten, von denen ein am jeweiligen Knoten endender Pfeil ausgeht, anzugeben. Für einen Schaltkreis der Größe t wären also t solche Einträge in die Liste notwendig. Um die Label der Knoten zu kodieren reicht, da in der Symbolmenge von $\mathcal{M}$ nur endlich viele Funktions- und Relationssymbole enthalten sind, und da der Schaltkreis konstantenfrei ist, für eine geeignete Konstante $B_\mathcal{M}$ ein Wort der Länge $B_\mathcal{M}$ aus. Wiederum aufgrund der Endlichkeit der Menge der Funktions- und Relationssymbole von $\mathcal{M}$ hat jeder der Knoten für eine geeignete Konstante $A_\mathcal{M}$ höchstens $A_\mathcal{M}$ eingehende Pfeile. Somit sind für jeden Knoten höchstens $A_\mathcal{M}$ Angaben zu den eingehenden Pfeilen notwendig, diese werden jeweils über die Angabe der Listennummer des sie aussendenden Knotens kodiert. Die Angaben zu den eingehenden Pfeilen eines Knotens können also in ein binäres Wort der Länge $A_\mathcal{M} \cdot \log(t)$ kodiert werden. Die Werte von $A_\mathcal{M}$ und $B_\mathcal{M}$ hängen dabei nur von der Struktur $\mathcal{M}$ ab. Insgesamt lässt sich also ein konstantenfreier Schaltkreis C der Größe t in ein binäres Wort der Länge $\mathcal{O}(t \cdot \log(t))$ kodieren.

Da Turingmaschinen über der Struktur $\mathcal{M}$ dieselben Funktionen und Relationen verwenden, wie konstantenfreie Schaltkreise über $\mathcal{M}$, lässt sich (analog zur Konstruktion einer universellen Turingmaschine) eine deterministische Turingmaschine über $\mathcal{M}$ konstruieren, welche bei Eingabe einer Kodierung eines konstantenfreien Schaltkreises C über $\mathcal{M}$ und eines Wortes $\bar{a} \in M^*$ der entsprechenden Länge den Wert $C(\bar{a})$ in polynomieller Zeit berechnet. Für die Beschreibung einer universellen Turingmaschine im klassischen Fall siehe zum Beispiel [21]. Dies führt sofort zu dem folgenden Ergebnis:

Proposition 2.1.6 (Auswertung von Schaltkreisen durch Turingmaschinen) *Sei $C(\bar{x})$ ein konstantenfreier Schaltkreis der Größe t über einer Struktur $\mathcal{M}$. Dann kann C in ein binäres Wort $w(C)$ der Länge $t \cdot C_\mathcal{M} \cdot \log(t)$ kodiert werden, wobei $C_\mathcal{M}$ nur von $\mathcal{M}$ abhängt. Die Funktion $(w(C), \bar{a}) \mapsto C(\bar{a})$ ist in polynomieller Zeit im Sinne von $\mathcal{M}$ berechenbar.*

□

Enthält die Symbolmenge von $\mathcal{M}$ nur endlich viele Konstantensymbole, so können auch diese jeweils in ein binäres Wort endlicher Länge kodiert werden, und Proposition

2.1.6 gilt für alle Schaltkreise über $\mathcal{M}$. Für eine Struktur mit überabzählbar vielen Konstanten ist es dagegen nicht mehr möglich, diese Konstanten in endliche binäre Folgen zu kodieren. In diesem Fall kodieren wir einen Schaltkreis $C(\bar{y})$ über $\mathcal{M}$, welcher mit den Konstantensymbolen $c_1, \ldots, c_k$ gelabelte Inputknoten enthält, indem wir zu dem Tupel $(C'(\bar{x}, \bar{y}), \bar{c})$ übergehen. Den Schaltkreis C' konstruieren wir, indem wir die k-vielen mit den Konstantensymbolen $c_1, \ldots, c_k$ gelabelten Inputknoten von C durch mit den Variablen $x_1, \ldots, x_k$ gelabelte Inputknoten ersetzen. Der Schaltkreis C' ist nun konstantenfrei und kann somit binär kodiert werden. Um für ein $\bar{a} \in M^*$ den Schaltkreis $C(\bar{a})$ auszuwerten, genügt es, den Schaltkreis $C'(\bar{c}, \bar{a})$ auszuwerten.

Schaltkreise können also mittels Turingmaschinen in polynomieller Zeit ausgewertet werden. Auf der anderen Seite läßt sich eine deterministische Turingmaschine (genauer: ihre Übergangsfunktion, die jeder Konfiguration die entsprechende Nachfolgekonfiguration zuordnet) jedoch auch durch eine Schaltkreisfolge darstellen. Da Schaltkreise nur mit Eingaben einer festen Länge rechnen, benötigen wir dazu folgende Definition:

Definition 2.1.7 (Konfiguration der Größe n) Sei $n \in \mathbb{N}$. Eine Konfiguration einer Turingmaschine *hat die Größe* n, falls sie auf jedem Band höchstens die Felder $-n$ bis n als nicht leer beschreibt und die Positionen der Zeiger auf $[-n, n]$ beschränkt sind.

Wenn eine Konfiguration einer Turingmaschine die Größe n hat, so hat sie auch die Größe n' für alle natürlichen Zahlen $n' > n$.

Da in einem Programmschritt die Größe einer Konfiguration höchstens (mittels einer Bewegung eines Zeigers) um eins erhöht werden kann, hat die Nachfolgekonfiguration einer Konfiguration der Größe n also höchstens die Größe $n + 1$. Im folgenden betrachten wir jedoch nur solche Konfigurationen der Größe n, deren Nachfolgekonfiguration ebenfalls die Größe n hat. Daß auch die Nachfolgekonfiguration die Größe n hat bedeutet dabei keine Einschränkung der Allgemeinheit, da eine Konfiguration der Größe n ja auch als eine Konfiguration der Größe $n + 1$ aufgefasst werden kann.

Der Begriff der Größe einer Konfiguration ermöglicht nun, Konfigurationen der Länge n auf eine kanonische Weise zu kodieren:

Definition 2.1.8 (Kanonische Kodierung einer Konfiguration) Sei $\mathbb{A}$ eine deterministische Turingmaschine über $\mathcal{M}$ mit k Bändern, und sei Q die Anzahl der Zeilen des Programms von $\mathbb{A}$. Die *kanonische Kodierung* einer Konfiguration K von $\mathbb{A}$ der Größe n ist ein Element $\bar{\varepsilon}_K \bar{a}_K$ aus $\{0,1\}^{\log(Q+1)+k\cdot(\log(n+1)+1)} \times M^{2k(2n+1)}$, welches die Konfiguration K auf folgende Weise darstellt:

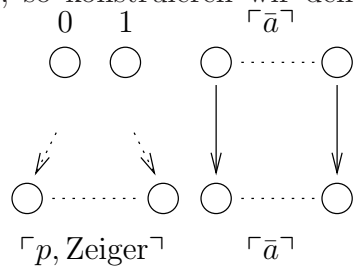
- Die ersten $\log(Q+1)$ Einträge des Tupels $\bar{\varepsilon}_K$ sind die binäre Kodierung der Zeilennummer der Konfiguration
- Die übrigen $k \cdot (\log(n+1)+1)$ Einträge von $\bar{\varepsilon}_K$ sind die binären Kodierungen der Zeigerpositionen der k Bänder. Hierbei werden jeweils in $\log(n+1)$ Bits der Betrag und in einem Bit das Vorzeichen der Nummer des durch den Zeiger markierten Feldes kodiert.
- Das Tupel $\bar{a}_K$ kodiert die Inhalte der insgesamt $k \cdot (2n+1)$ Felder. Hierbei wird jedes Feld durch 00 kodiert, falls es leer ist, und durch $1a$, falls a der Inhalt des Feldes ist.

Bevor wir die Übergangsfunktion einer Turingmaschine generell durch eine Schaltkreisfolge darstellen, betrachten wir zunächst den einfacheren Fall, in dem die Zeilennummer und die Positionen der Zeiger fest sind.

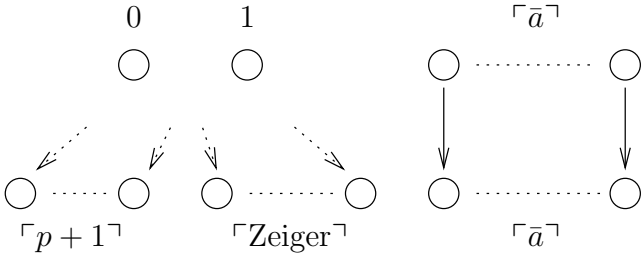
Lemma 2.1.9 *Sei $\mathbb{A}$ eine deterministische Turingmaschine über $\mathcal{M}$ mit k Bändern, und sei Q die Anzahl der Zeilen des Programmes von $\mathbb{A}$. Sei für ein $n \in \mathbb{N}$ das Element $\bar{\varepsilon} \in \{0,1\}^{\log(Q+1)+k(\log(n+1)+1)}$ die Kodierung einer Zeilennummer und der Positionen der k Zeiger. Dann gibt es einen Schaltkreis $C_{\bar{\varepsilon}}$ linearer Größe in n, welcher $2k(2n+1)$ mit Variablen gelabelte Eingabeknoten und $\log(Q+1)+k(\log(n+1)+1)+2k(2n+1)$ Ausgabeknoten hat und bei Eingabe von $\bar{x} \in M^{2k(2n+1)}$ die kanonische Kodierung der Nachfolgekonfiguration der durch $\bar{\varepsilon}\bar{x}$ kodierten Konfiguration ausgibt, falls diese Nachfolgekonfiguration die Größe n hat.*

Beweis: Wir betrachten zunächst die möglichen Befehle und konstruieren für jeden dieser Befehle einen eigenen Schaltkreis. Diese fügen wir zuletzt zu einem einzigen großen Schaltkreis zusammen. Jeder der kleinen Schaltkreise hat zwei mit 0 und 1 gelabelte und $2k(2n+1)$ mit Variablen gelabelte Eingabeknoten und $\log(Q+1)+k(\log(n+1)+1)+2k(2n+1)$ Ausgabeknoten, welche die nächste Zeilennummer, die Positionen der Zeiger und die Inhalte der Felder kodieren. Der Übersichtlichkeit halber seien alle Knoten falls nicht anders angegeben mit Id gelabelt.

Falls der Befehl ein Stopbefehl ist, falls also $\Pi_p =$`(stop)`, so konstruieren wir den Schaltkreis wie folgt: Von den mit 0 und 1 gelabelten Eingabeknoten gehen Pfeile zu den Ausgabeknoten, welche die nächste Zeilennummer und die Zeigerpositionen kodieren, so daß als nächste Zeile gerade p kodiert und die Zeigerpositionen beibehalten werden. Von den mit Variablen gelabelten Eingabeknoten gehen Pfeile zu den Ausgabeknoten, die die Inhalte der Felder kodieren, so daß die Inhalte der Felder gleich bleiben.

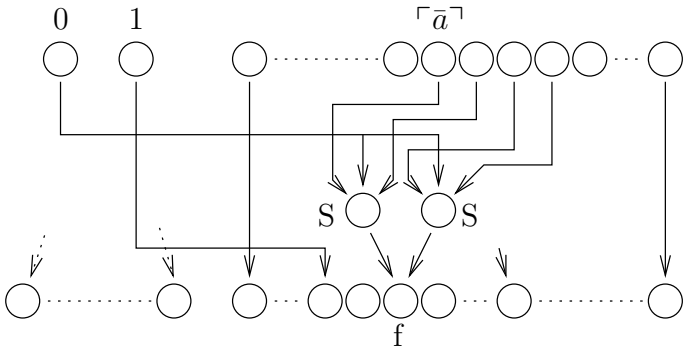


Falls $\Pi_p =$`(move` i `right)` oder $\Pi_p =$`(move` i `left)`, so werden mit Pfeilen von mit 0 und 1 gelabelten Knoten die Kodierung der Zeilennummer $p+1$ und die Kodierung der neuen Zeigerpositionen erstellt, wobei natürlich die Position des i-ten Zeigers dem Befehl entsprechend verändert wird. Die mit Variablen $\bar{a}$ gelabelten Eingangsknoten werden identisch auf die entsprechenden Ausgangsknoten abgebildet. Der Fall, daß die Position eines Zeigers nicht mehr kodiert werden kann, weil dieser zu weit nach rechts oder links bewegt wurde, braucht uns hier keine Sorgen zu machen, da wir ja nur solche Konfigurationen der Größe n betrachten, deren Nachfolgekonfigurationen ebenfalls die Größe n haben.

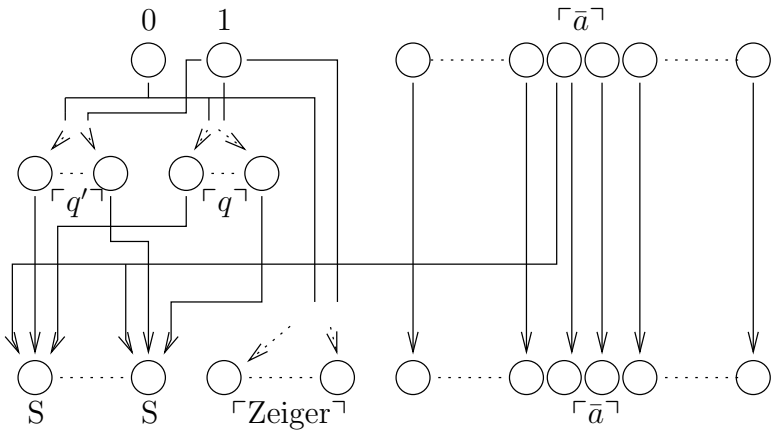


Falls $\Pi_p =$`(compute` f, i `to` j`)`, werden wieder zunächst mithilfe von Pfeilen von den mit 0 und 1 gelabelten Eingabeknoten die Zeilennummer $p+1$ und die Zeigerpositionen kodiert. Außerdem werden alle mit Variablen gelabelte Eingabeknoten bis auf die zwei, welche den Inhalt des durch den Zeiger des j-ten Bandes markierten Feldes kodieren, identisch auf die entsprechenden Ausgabeknoten abgebildet. Mit einem Pfeil von dem

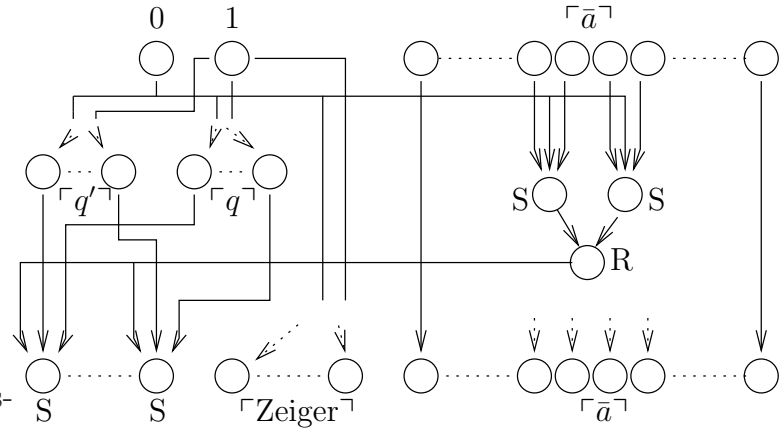
mit 1 gelabelten Eingabeknoten zu demjenigen Ausgabeknoten, der kodiert, ob das durch den Zeiger des j-ten Bandes markierte Feld leer ist, wird gekennzeichnet, daß dieses nun nicht leer ist. Schließlich wird der Ausgabeknoten, der den Inhalt des den Zeiger des j-ten Bandes markierten Feldes angibt, mit der Funktion f gelabelt und erhält eingehende Pfeile von denjenigen Eingabeknoten, welche die Inhalte der entsprechenden Felder kodieren, wobei jeweils ein Selektorknoten dazwischengeschaltet wird, der sicherstellt, daß der entsprechende Feldinhalt verwendet wird, falls dieses nicht leer ist, und 0 sonst. Sollte f auf die Inhalte von außerhalb der Konfiguration liegenden Feldern angewand werden, so erhält der mit f gelabelte Knoten statt von den entsprechenden mit Variablen gelabelten Eingabeknoten Pfeile von dem mit 0 gelabelten Knoten. Das Diagramm zeigt das Beispiel einer zweistelligen Funktion.



Falls $\Pi_p =$`(if` $i = \lambda$ `then` q`, else` q'`)`, so bilden wir mittels Pfeilen von den mit 0 und 1 gelabelten Knoten zunächst die Kodierungen der Zeilennummern q und q'. Die die neue Zeilennummer kodierenden Ausgangsknoten sind mit dem Selektor gelabelt und erhalten ihren ersten Pfeil von demjenigen Eingangsknoten, der kodiert, ob das durch den Zeiger des i-ten Bandes markierte Feld leer ist. Ihren zweiten und dritten Pfeil erhalten sie von dem entsprechenden Knoten der Kodierung von q beziehungsweise q'. Die Zeigerpositionen werden gleichbleibend kodiert und die mit Variablen gelabelten Eingabeknoten identisch auf die entsprechenden Ausgabeknoten abgebildet.



Falls $\Pi_p =$`(if` i, R `then` q`, else` q'`)`, verfahren wir ähnlich wie im letzten Fall: die mit Variablen gelabelten Eingabeknoten werden identisch auf die entsprechenden Ausgabeknoten abgebildet, und mit Pfeilen von den mit 0 und 1 gelabelten Knoten werden die Zeigerpositionen unverändert kodiert. Die mit dem Selektor gelabelten Ausgangsknoten, welche die neue



Zeilennummer kodieren werden, erhalten ihren ersten Pfeil diesmal jedoch von einem mit χ_R gelabelten Knoten. Dieser wiederum erhält der Stelligkeit von R gemäß seine Pfeile von denjenigen Eingabeknoten, welche das durch den Zeiger des i-ten Bandes markierte Element und die nächsten Elemente kodieren. Um auch den Fall, daß eines der Felder leer sein sollte, zu beachten, werden jeweils noch mit dem Selektor gelabelte Knoten dazwischengeschaltet, welche ihren ersten Pfeil von demjenigen Knoten erhalten, der kodiert, ob das jeweilige Feld leer ist, ihren zweiten Pfeil von dem mit 0 gelabelten Knoten und ihren dritten Pfeil schließlich von dem den Inhalt des jeweiligen Feldes kodierenden Eingabeknoten. Sollten außerhalb der Konfiguration der entsprechenden Größe liegende Felder benötigt werden, so erhält der mit χ_R gelabelte Knoten die entsprechenden Pfeile von dem mit 0 gelabelten Knoten. Das Diagramm zeigt die Konstruktion anhand des Beispiels einer zweistelligen Relation.

In jedem dieser Fälle hat der konstruierte Schaltkreis $2 + 2k(2n + 1) + \log(Q + 1) + k(\log(n + 1) + 1) + 2k(2n + 1)$ Ein- und Ausgabeknoten. Für die Fälle $\Pi_p =$`(stop)` und $\Pi_p =$`(move` i `right)` beziehungsweise $\Pi_p =$`(move` i `left)` kommen keine weiteren Knoten hinzu, für $\Pi_p =$`(compute` f, i `to` j`)` wird abhängig von der Stelligkeit der Funktion eine nicht von n abhängige Anzahl von Knoten hinzugefügt. Für $\Pi_p =$`(if` $i = \lambda$ `then` q`, else` q'`)` kommen $2\log(Q + 1)$ Knoten hinzu und für $\Pi_p =$`(if` i, R `then` q`, else` q'`)` die Anzahl von $2\log(Q+1)+K+1$ Knoten, wobei K die Stelligkeit der Relation ist. Da die Anzahl der möglichen Programmzeilen und die Stelligkeiten der Funktionen und Relationen nicht von n, also der Größe der Konfiguration, abhängig sind, ist die Größe der konstruierten Schaltkreise insgesamt also sogar linear in n. □

Um den Schaltkreis zu konstruieren, der aus der kanonischen Kodierung einer beliebigen Konfiguration der Größe n die kanonische Kodierung ihrer Nachfolgekonfiguration berechnet, benötigen wir der Übersichtlichkeit halber noch ein Hilfsmittel zur Zusammenführung aller möglichen Schaltkreise der Art, wie sie im letzten Lemma konstruiert wurden:

Definition 2.1.10 (Giga-Selektor) Für eine Struktur $\mathcal{M}$ und $N \in \mathbb{N}_{>0}$ ist ein *Giga-Selektor* $S_N : \{0,1\}^N \times M^{2^N} \to M$ eine $N + 2^N$-stellige Funktion, für die gilt:

$$S(\varepsilon_1, \ldots, \varepsilon_N, y_{\underbrace{0\ldots00}_{N\text{-mal}}}, y_{0\ldots01}, \ldots, y_{1\ldots11}) = y_{\varepsilon_1,\ldots,\varepsilon_N}$$

.

Für $N = 1$ erhalten wir so den schon bekannten Selektor. Desweiteren ist S_2 durch S_1 darstellbar als $S_2(x_1, x_2, y_{00}, y_{01}, y_{10}, y_{11}) = S(x_1, S(x_2, y_{00}, y_{01}), S(x_2, y_{10}, y_{11}))$. Durch Iteration dieses Schrittes erhalten wir die allgemeinere Darstellung

$$S_{N+1}(x_1, \ldots, x_N, x_{N+1}, y_{\underbrace{0\ldots00}_{(N+1)\text{-mal}}}, y_{0\ldots01}, \ldots, y_{\bar{\varepsilon}0}, y_{\bar{\varepsilon}1}, \ldots)$$

$$= S_N(x_1, \ldots, x_N, S(x_{N+1}, y_{0\ldots00}, y_{0\ldots01}), \ldots, S(x_{N+1}, y_{\bar{\varepsilon}0}, y_{\bar{\varepsilon}1}), \ldots)$$

Ein mit S_N gelabelter Knoten in einem Schaltkreis läßt sich also durch einen Subschaltkreis der Größe $\mathcal{O}(2^N)$ darstellen.

Damit stehen alle Hilfsmittel zur Verfügung, um einen Schaltkreis zu konstruieren, der bei Eingabe einer kanonischen Kodierung einer Konfiguration einer beschränkten Größe deren Nachfolgekonfiguration berechnet:

Satz 2.1.11 (Darstellung der Übergangsfunktion durch einen Schaltkreis) *Sei $\mathbb{A}$ eine deterministische Turingmaschine über $\mathcal{M}$ mit k Bändern, und sei Q die Anzahl der Zeilen des Programmes von $\mathbb{A}$. Sei weiterhin f_n diejenige Funktion, welche die kanonische Kodierung jeder Konfiguration der Größe n, deren Nachfolgekonfiguration ebenfalls die Größe n hat, auf die kanonische Kodierung ebendieser Nachfolgekonfiguration abbildet. Dann ist f_n durch einen Schaltkreis im Sinne von $\mathcal{M}$ von polynomieller Größe in n darstellbar.*

Beweis: Sei n eine natürliche Zahl größer null. Wir konstruieren zunächst mit dem vorhergehenden Lemma jeden möglichen Schaltkreis für eine Konfiguration von $\mathbb{A}$ der Größe n. All diese Schaltkreise werden dann mit Hilfe von Giga-Selektoren in dem gesuchten Schaltkreis C zusammengefügt. Da die Kodierung der Zeilennummer und der Zeigerpositionen die Länge $\log(Q+1) + k(\log(n+1)+1)$ hat, gibt es $2^{\log(Q+1)+k(\log(n+1)+1)} =: N$ mögliche Kodierungen $\bar{\varepsilon}$ von Kombinationen von Zeilennummern und Zeigerpositionen und somit auch N zugehörige Schaltkreise $C_{\bar{\varepsilon}}$.

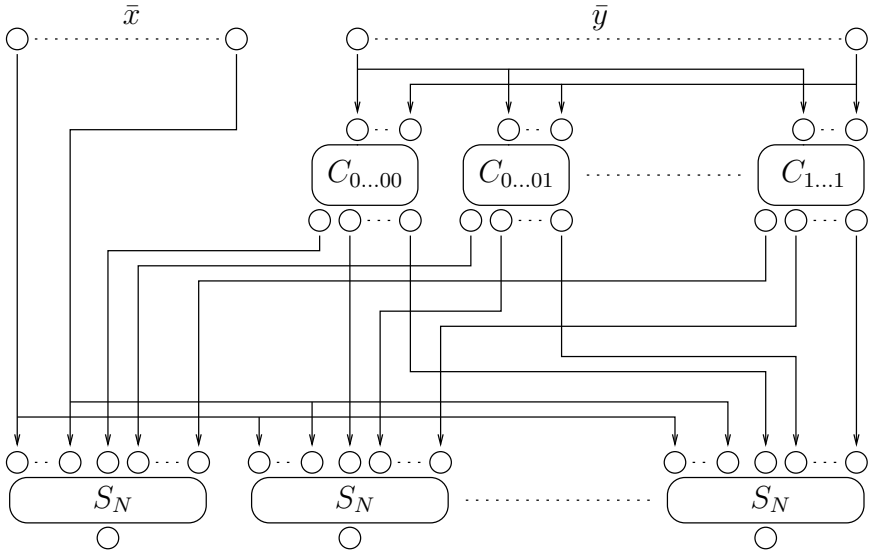


Abbildung 2.1: Der die Nachfolgekonfiguration einer Konfiguration der Größe n berechnende Schaltkreis C

Der Schaltkreis C hat nun $N+2k(2n+1)$ Eingabeknoten, von denen die ersten N Stück für die Eingabe der Kodierung der Zeilennummer und der Zeigerpositionen zuständig sind, die übrigen $2k(2n+1)$ Knoten für die Eingabe der Feldinhalte. Die die Feldinhalte kodierende Eingabeknoten verbinden wir mit den entsprechenden Eingabeknoten aller Subschaltkreise $C_{\bar{\varepsilon}}$. Auf diese Weise berechnen die Subschaltkreise $C_{\bar{\varepsilon}}$ zunächst alle möglichen nächsten Konfigurationen. Um aus diesen die zur Eingabe gehörige Nachfolgekonfiguration auszuwählen, benutzen wir nun Giga-Selektoren S_N, deren erste N Eingänge jeweils

Pfeile von den die Zeilennummer und die Zeigerpositionen kodierenden Eingabeknoten unseres Schaltkreises C empfangen. Desweiteren erhält der erste dieser Giga-Selektoren die folgenden Inputs jeweils von den ersten Ausgangsknoten der Subschaltkreise $C_{\bar{\varepsilon}}$. Die Reihenfolge ist dabei gemäß der lexikographischen Ordnung auf $\{0,1\}^N$. Den ersten dieser Inputs erhält der erste Giga-Selektor also von dem Subschaltkreis $C_{0\ldots00}$, den zweiten von $C_{0\ldots,01}$, den letzten von $C_{1\ldots11}$. Analog erhält der i-te Giga-Selektor die entsprechenden Inputs von den i-ten Ausgangsknoten der Subschaltkreise $C_{\bar{\varepsilon}}$. Somit wählt der i-te Giga-Selektor die i-te Ausgabe des zu der eingegebenen Kodierung der Zeilennummer und Zeigerpostitionen gehörigen Subschaltkreises aus. Für eine graphische Darstellung dieses Schrittes der Konstruktion des Schaltkreises C siehe Abbildung 2.1.

In einem letzten Schritt werden die Giga-Selektoren durch die entsprechenden nur den einfachen Selektor verwendenden Subschaltkreise ersetzt. Da es nur $2^{\log(Q+1)+k(\log(n+1)+1)} = \mathcal{O}(n)$ viele Subschaltkreise $C_{\bar{\varepsilon}}$ gibt, die nach dem vorherigen Lemma alle in der Größe polynomiell in n sind, und da die Größe der die $\mathcal{O}(n)$ vielen Giga-Selektoren darstellenden Subschaltkreise ebenfalls polynomiell in n ist, ist schließlich auch die Größe von C polynomiell in n. □

2.1.2 Die Klassen $\mathrm{P}_{\mathcal{M}}$ und $\mathrm{EXP}_{\mathcal{M}}$

Mithilfe der deterministischen Turingmaschinen über einer Struktur $\mathcal{M}$ lassen sich nun schon wichtige Komplexitätsklassen über der Struktur $\mathcal{M}$ definieren. Ähnlich wie bei den Schaltkreisen betrachten wir Probleme als „handhabbar“, falls der das Problem entscheidende Algorithmus „nicht zu lange“ läuft, falls also die Laufzeit in Abhängigkeit von der Länge der Eingabe nicht allzu groß wird:

Definition 2.1.12 ($\mathrm{P}_{\mathcal{M}}$) Ein Problem $X \subseteq M^*$ ist in der Klasse $\mathrm{P}_{\mathcal{M}}$, falls die charakteristische Funktion von X in polynomieller Zeit im Sinne von $\mathcal{M}$ berechenbar ist.

Mit Satz 2.1.11 ergibt sich eine Charakterisierung der Klasse $\mathrm{P}_{\mathcal{M}}$ mit Hilfe des Schaltkreiskonzeptes:

Satz 2.1.13 *Sei $\mathcal{M}$ eine Struktur. Ein Problem $X \subseteq M^*$ ist in der Klasse $\mathrm{P}_{\mathcal{M}}$ genau dann, wenn es eine das Problem X akzeptierende gesetzte Familie $(C_n)_{n\in\mathbb{N}_{>0}}$ von Schaltkreisen über $\mathcal{M}$ gibt, so daß die Funktion $1^n \mapsto \ulcorner C_n \urcorner$, welche eine natürliche Zahl n in unärer Darstellung auf eine Kodierung des Schaltkreises C_n abbildet, in polynomieller Zeit im Sinne von $\mathcal{M}$ berechenbar ist.*

Beweis: “$\leftarrow$“: Sei also $(C_n)_{n\in\mathbb{N}_{>0}}$ eine derartige Familie von Schaltkreisen. Eine das Problem X in polynomieller Zeit entscheidende deterministische Turingmaschine im Sinne von $\mathcal{M}$ berechnet bei Eingabe eines Elements $\bar{x} \in M^*$ nun zunächst in $p_1(|\bar{x}|)$ vielen Schritten die Kodierung des Schaltkreises $C_{|\bar{x}|}$. Diese ist dann offensichtlicherweise höchstens $p_1(|\bar{x}|)$ viele Zeichen lang. Dann berechnet sie in höchstens $p_2(|\ulcorner C_{|\bar{x}|} \urcorner|) + |\bar{x}|) \leq p_2(p_1(|\bar{x}|) + |\bar{x}|)$ vielen Schritten den Wert $C_{|\bar{x}|}(\bar{x})$ und gibt diesen aus. Insgesamt benötigt sie also nur polynomiell viele Schritte.

„$\rightarrow$“: Sei $X \in \mathrm{P}_{\mathcal{M}}$ und $\mathbb{A}$ die das Problem X in Zeit p entscheidende deterministische Turingmaschine über $\mathcal{M}$ mit k Bändern. Da $\mathbb{A}$ bei Eingabe von $\bar{x} \in M^n$ höchstens $p(n)$

Schritte läuft, somit also höchstens $p(n)$ Felder ihrer Bänder beschreiben kann, haben alle durchlaufenen Konfigurationen von $\mathbb{A}$ bei Eingabe von $\bar{x}$ die Größe $p(n)$. Nach Satz 2.1.11 ist die Übergangsfunktion von $\mathbb{A}$ für Konfigurationen der Größe $p(n)$ durch einen Schaltkreis $C'_{p(n)}$ der Größe $p'(n)$ darstellbar. Desweiteren ist es mit dem in den Beweisen von Lemma 2.1.9 und Satz 2.1.11 verwendeten Konstruktionsverfahren offensichtlich möglich, den Schaltkreis $C_{p(n)}$ in polynomieller Zeit zu konstruieren.

Als nächstes werden $p(n)$ Kopien des Schaltkreises $C'_{p(n)}$ hintereinandergeschaltet. Der so erzeugte Schaltkreis simuliert dann eine $p(n)$-malige Anwendung der Übergangsfunktion auf die kanonische Kodierung einer Konfiguration. Nun müssen wir nur noch mittels Pfeilen von zwei mit 0 und 1 gelabelten Knoten bei den Eingangsknoten der ersten Kopie von $C'_{p(n)}$ die Kodierung einer Startkonfiguration fest verdrahten. Da der neue Schaltkreis außerdem nur diejenigen Eingabeknoten haben soll, die die Feldinhalte der ersten n Felder des ersten Bandes kodieren, erhalten die Eingabeknoten, die kodieren, ob das entsprechende Feld leer ist, einen Pfeil von dem mit 1 gelabelten Knoten. Die Eingabeknoten, die die Inhalte der übrigen Felder kodieren, erhalten einen Pfeil von dem mit 0 gelabelten Knoten. Somit sind sie auch als leer gekennzeichnet.

Der so konstruierte Schaltkreis gibt nun bei Eingabe von $\bar{x} \in M^n$ noch nicht das Ergebnis der Berechnung von $\mathbb{A}$ aus, sondern die ganze Haltekonfiguration. Um nur das Ergebnis der Berechnung von $\mathbb{A}$ auszugeben, werden die Ausgabeknoten mittels mit dem Selektor gelabelten Knoten derart in einen einzigen Ausgabeknoten zusammengefasst, daß dieser den Inhalt des ersten Feldes des Ausgabebandes ausgibt. Dies ist möglich, da $\mathbb{A}$ ja auf jeden Fall 0 oder 1 ausgibt, und da somit derjenige Ausgabeknoten, der kodiert, ob dieses Feld leer ist, auf jeden Fall 1 ausgibt. Da für die Elimination eines überflüssigen Ausgabeknotens ein Selektorknoten hinzukommt, werden also auch nur polynomiell viele neue Selektorknoten benötigt.

Die so konstruierte Schaltkreisfamilie ist gesetzt, da die Maschine $\mathbb{A}$ nur endlich viele Konstanten $c_1, \ldots, c_k$ aus $\mathcal{M}$ verwendet, und somit auch jeder der oben konstruierten Schaltkreise nur die Konstanten $c_1, \ldots, c_k$ verwendet. □

Da eine gesetzte Schaltkreisfamilie, die in polynomieller Zeit berechenbar ist, sicherlich auch existiert, folgt damit sofort:

Korollar 2.1.14 *Für jede Struktur $\mathcal{M}$ gilt* $\mathrm{P}_{\mathcal{M}} \subseteq \mathbb{P}_{\mathcal{M}}$.

□

Andererseits muß eine existierende Schaltkreisfolge noch nicht schnell berechenbar sein, im allgemeinen gilt nicht $\mathrm{P}_{\mathcal{M}} = \mathbb{P}_{\mathcal{M}}$:

Lemma 2.1.15 *Für Strukturen mit höchstens abzählbarer Symbolmenge gilt:* $\mathrm{P}_{\mathcal{M}} \neq \mathbb{P}_{\mathcal{M}}$

Beweis: In einer Struktur $\mathcal{M}$ mit abzählbarer Symbolmenge kann jede deterministische Turingmaschine binär kodiert werden. Somit gibt es höchstens $\aleph_0$-viele deterministische Turingmaschinen über $\mathcal{M}$ und also auch nur höchstens $\aleph_0$-viele Probleme in $\mathrm{P}_{\mathcal{M}}$. Andererseits gibt es nach Bemerkung 1.1.9 mindestens $2^{\aleph_0}$ viele Probleme in $\mathbb{P}_{\mathcal{M}}$. □

Der Beweis des vorhergehenden Lemmas stützt sich wesentlich darauf, daß in unserer Definition einer Turingmaschine nur der Gebrauch von solchen Parametern, die in der Menge der Konstanten der Struktur vorkommen, erlaubt wurde. Betrachtet man Strukturen mit genügend Konstanten, in die die Informationen über die möglichen Schaltkreisfolgen kodiert werden können, so ändert sich die Situation:

Satz 2.1.16 *In der Struktur* $\mathcal{R} = (\mathbb{R}, +, -, <, (c_r)_{r\in\mathbb{R}})$ *gilt:* $\mathrm{P}_{\mathcal{R}} = \mathbb{P}_{\mathcal{R}}$.

Beweis: Nach Bemerkung 2.1.14 gilt $\mathrm{P}_{\mathcal{M}} \subseteq \mathbb{P}_{\mathcal{M}}$, es ist also nur die Rückrichtung zu zeigen. Sei hierfür $X \subseteq \mathbb{R}^*$ ein Problem in $\mathbb{P}_{\mathcal{R}}$, und sei $(C_n)_{n\in\mathbb{N}_{>0}}$ die entsprechende gesetzte Schaltkreisfamilie welche die Konstanten $c_1, \ldots, c_k$ verwendet und das Problem X akzeptiert. Nach Definition von $\mathbb{P}_{\mathcal{R}}$ gilt dann $t(C_n) \leq p(n)$ für ein Polynom p. Die den Schaltkreisen C_i entsprechenden konstantenfreien Schaltkreise C'_i können jeweils binär kodiert werden. Somit lässt sich auch die gesamte Schaltkreisfamilie $(C'_n)_{n\in\mathbb{N}_{>0}}$ binär kodieren, indem man zum Beispiel im Code der einzelnen Schaltkreise ein Zeichen x durch $1x$ ersetzt und dann die modifizierten Codes der Schaltkreise getrennt durch das Trennungszeichen 00 aneinanderreiht. Die so gewonnene binäre Folge $b_1, b_2, b_3, \ldots$ lässt sich nun in die reelle Zahl $r = \sum_{i=1}^{n} b_i \cdot 2^{-i}$ übersetzen. Da die Codes zweier Schaltkreise jeweils durch das Trennungszeichen 00 getrennt sind, sind auch solche binären Folgen ausgeschlossen, die irgendwann konstant 1 werden. Dies verhindert, daß r gleichzeitig zwei verschiedene Schaltkreisfamilien bezeichnet.

Um den Code eines bestimmten Schaltkreises C' aus der Zahl r zu gewinnen, benötigt man die entsprechenden Ziffern der Binärentwicklung von r. Diese erhält man jedoch mittels der beiden Funktionen

$$f : \mathbb{R} \to \{0,1\},\ f(x) = \begin{cases} 0, & \text{falls } 2 \cdot x < 1 \\ f(x) = 1, & \text{sonst} \end{cases}$$

und

$$g : \{0,1\} \to \{0,1\},\ g(x) = 2 \cdot x - f(x).$$

Die i-te Ziffer der Binärentwicklung von r ist dann gerade

$$b_i = \overbrace{g(g(\ldots g}^{(i-1)\text{-mal}}(f(r))\ldots))$$

Da die Funktionen f und g mit den Funktionen der Struktur $\mathcal{R}$ berechnet werden können, gibt es eine Konstante A, so daß eine deterministische Turingmaschine im Sinne von $\mathcal{R}$ die Ziffer b_i in höchstens $A \cdot i$ Schritten aus r berechnen kann. Da nun die Größe der Schaltkreise durch das Polynom p beschränkt ist, benötigt man zu Rekonstruktion des Codes des Schaltkreises C'_n nur die ersten $2 \cdot (p(1) + 1 + p(2) + 1 + p(3) + \ldots + 1 + p(n))$ Ziffern.

Die deterministische Turingmaschine im Sinne der Struktur $\mathcal{R}$, die das Problem X entscheidet, berechnet also bei Eingabe eines Elementes $\bar{x} \in \mathbb{R}^*$ in höchstens $A \cdot (2 \cdot |\bar{x}| \cdot (p(|\bar{x}|) + 1)^2)$ Schritten den Code des Schaltkreises $C'_{|\bar{x}|}(\bar{w}, \bar{x})$, und wertet $C'_{|\bar{x}|}(\bar{c}, \bar{x}) = C_{|\bar{x}|}(\bar{x})$ dann (nach Satz 2.1.6 in polynomieller Zeit) aus. Somit ist X in $\mathrm{P}_{\mathcal{M}}$. □

Da wir nur Strukturen betrachten, deren Symbolmenge nur endlich viele mehr als nullstellige Funktions- und Relationssymbole enthält, und in denen somit jedes dieser Symbole endlich binär kodiert werden kann, gilt sogar:

Korollar 2.1.17 *In jeder Struktur* $\mathcal{M} = (M, f_1, \ldots, f_n, R_1, \ldots, R_m, (c_i)_{i\in I})$ *mit* $\{c_r \mid r \in \mathbb{R}\} \subseteq \{c_i \mid i \in I\}$ *und* $(\mathbb{R}, +, -, <, (c_r)_{r\in\mathbb{R}}) \subseteq \mathcal{M} \restriction_{\{+,-,<,(c_r)_{r\in\mathbb{R}}\}}$ *gilt:* $\mathrm{P}_{\mathcal{M}} = \mathbb{P}_{\mathcal{M}}$.

□

Bemerkung 2.1.18 (Alternative Charakterisierung für $\mathbb{P}_{\mathcal{M}}$) Die Klasse $\mathbb{P}_{\mathcal{M}}$ ließe sich alternativ auch charakterisieren als die Klasse der Probleme, welche in polynomieller Zeit durch *Turingmaschinen mit binärem advice* entscheidbar sind. Eine Turingmaschine mit binärem advice ist eine deterministische Turingmaschine, welche bei Eingabe eines Elementes zusätzlich auf einen nur von der Länge der Eingabe abhängenden binären advice zurückgreifen kann. Der advice ist dabei in der Form einer 0-1-Folge gegeben, deren Länge polynomiell in der Länge der Eingabe ist. Dieser advice kann als Kodierung des entsprechenden konstantenfreien Schaltkreises aus unserer Definition von $\mathbb{P}_{\mathcal{M}}$ gesehen werden.

Bemerkung 2.1.19 (Alternative Charakterisierung für $\mathbb{P}^*_{\mathcal{M}}$) Ebenso läßt sich die Klasse $\mathbb{P}^*_{\mathcal{M}}$ charakterisien als die Klasse der Probleme, welche in polynomieller Zeit durch *Turingmaschinen mit advice in* M^* entscheidbar sind. Der advice lässt sich hierbei als Kodierung des entsprechenden Schaltkreises mit Angabe der von ihm verwendeten Konstanten verstehen.

Eine andere Komplexitätsklasse, die bei der Behandlung der nichtdeterministischen Turingmaschinen über einer Struktur $\mathcal{M}$ noch eine Rolle spielen wird, ist die Klasse $\mathrm{EXP}_{\mathcal{M}}$, die wir an dieser Stelle nur kurz einführen, jedoch nicht eingehender betrachten:

Definition 2.1.20 ($\mathrm{EXP}_{\mathcal{M}}$) Ein Problem $X \subseteq M^*$ ist in der Klasse $\mathrm{EXP}_{\mathcal{M}}$, falls die charakteristische Funktion von X in exponentieller Zeit im Sinne von $\mathcal{M}$ berechenbar ist.

Da man für jedes Polynom p einen Faktor $C \in \mathbb{R}$ und ein Polynom q finden kann, so daß für alle natürlichen Zahlen n gilt: $p(n) \leq C \cdot \exp(q(n))$, ist jedes in polynomieller Zeit entscheidbare Problem auch in exponentieller Zeit entscheidbar. Es gilt also $\mathrm{P}_{\mathcal{M}} \subseteq \mathrm{EXP}_{\mathcal{M}}$.

Im folgenden wird sich herausstellen, daß eine bestimmte Art von Nichtdeterminismus mit der Klasse $\mathrm{EXP}_{\mathcal{M}}$ in Zusammenhang steht. Doch zunächst zu den nichtdeterministischen Maschinen.

2.2 Nichtdeterministische Turingmaschinen

Der Begriff des Nichtdeterminismus wird in diesem Zusammenhang so verstanden, daß es einer Maschine über einer Struktur erlaubt ist, Elemente dieser Struktur zu raten. Wir werden vollständig nichtdeterministische Maschinen definieren, denen genau dies erlaubt ist. Da andererseits jede der hier betrachteten Strukturen Elemente 0 und 1 enthält, ist es überdies interessant, eine andere Art von Nichtdeterminismus, den binären Nichtdeterminsmus, zu betrachten. Einer binär nichtdeterministischen Maschine ist nur das Raten von Elementen aus der Menge $\{0, 1\}$ erlaubt. Zunächst zu den binär nichtderministischen Maschinen.

2.2.1 Binär nichtdeterministische Turingmaschinen

Definition 2.2.1 (Binär nichtdeterministische Turingmaschine über $\mathcal{M}$) Eine *binär nichtdeterministische Turingmaschine über* $\mathcal{M}$ ist eine deterministische Turingmaschine über $\mathcal{M}$, in deren Programm zusätzlich zu den deterministischen Befehlen

- `(move` i `right)` oder `(move` i `left)`
- `(compute` f`,`i `to` j`)`
- `(if` R`,`i `then` p`, else` q`)`
- `(if` $i = \lambda$ `then` p`, else` q`)`
- `(stop)`

mit den gewohnten Bedeutungen auch noch binäre Ratebefehle vorkommen können. Ein binärer Ratebefehl hat die Form

- `(bin-guess` i`)`,

mit $i \in [k]$. Wenn an einer Stelle der Berechnung einer binär nichtdeterministischen Turingmaschine ein solcher binärer Ratebefehl auftaucht, so wird ein Element aus $\{0, 1\}$ geraten und in das vom Zeiger des i-ten Bandes markierte Feld geschrieben. Die verschiedenen Arten von *Konfigurationen* und der Begriff der *Berechnung* werden analog wie im deterministischen Fall definiert.

Bemerkung 2.2.2 (Übergangsrelation) Da bei Verwendung von binären Ratebefehlen die Nachfolgekonfiguration einer gegebenen Konfiguration nicht mehr eindeutig bestimmt sein muß, kann man nicht mehr von einer Übergangsfunktion der Maschine sprechen. Stattdessen erhält man nun den Begriff der *Übergangsrelation* der Maschine: diese beschreibt, welche Konfigurationen aus einer gegebenen Konfiguration hervorgehen können.

Da also auch nicht mehr alle möglichen Berechnungen einer binär nichtdeterministischen Turingmaschine angesetzt auf eine bestimmten Eingabe dasselbe Ergebnis liefern müssen, ist in diesem Zusammenhang der Begriff einer eine Menge entscheidenden Maschine nicht mehr zweckmäßig. Stattdessen führen wir das analoge Konzept einer eine Menge akzeptierenden Maschine ein:

Definition 2.2.3 Sei $\mathcal{M}$ eine Struktur mit Universum M und $\mathbb{A}$ eine binär nichtdeterministische Turingmaschine über $\mathcal{M}$. Eine ein Element $\bar{a} \in M^*$ *akzeptierende Berechnung* von $\mathbb{A}$ ist eine Berechnung mit der die Eingabe $\bar{a}$ enthaltenden Startkonfiguration, deren Haltekonfiguration als Ausgabe das Element 1 bestimmt. Die binär nichtdeterministische Turingmaschine $\mathbb{A}$ *akzeptiert die Sprache* $L \subseteq M^*$, falls es für jedes Element aus L mindestens eine und für Elemente aus $M^* \setminus L$ keine akzeptierende Berechnung von $\mathbb{A}$ bei Eingabe des entsprechenden Elementes gibt.

Wichtig hierbei ist, daß es für Elemente der Sprache L nur mindestens *eine* akzeptierende Berechnung geben muß, also diese eine Folge von Programmschritten irgendwann eine Halteanweisung beinhaltet. Die anderen möglichen Durchläufe der Maschine können durchaus irgendwann in eine Endlosschleife gehen, müssen also nicht irgendwann anhalten. Hierzu ein einfaches Beispiel:

Beispiel 2 (NULLSACK) Sei $\mathcal{M} = (\mathbb{Z}, +)$. Nach unserer Konvention lassen wir wieder die Konstantensymbole $0, 1$ und die Funktions- beziehungsweise Relationssymbole $\mathrm{Id}, S, =$ in der Angabe der Symbolmenge von $\mathcal{M}$ weg. Wir betrachten das Problem

NULLSACK
Eingabe: $\bar{z} = z_1 \dots z_n \in \mathbb{Z}^*$
Frage: Gibt es ein $I \subseteq [n]$ mit $\sum_{i \in I} z_i = 0$?

NULLSACK ist also das Problem, bei endlich vielen gegebenen ganzen Zahlen $z_1, \dots, z_n$ zu entscheiden, ob sich aus ihnen nur durch Addition die Null gewinnen läßt. Betrachten wir eine binär nichtdeterministische Turingmaschine $\mathbb{A}$ über $\mathcal{M}$ mit vier Bändern und dem folgenden Programm:

(1,`(bin-guess 3)`)
(2,`(move 3 left)`)
(3,`(compute 0 to 3)`)
(4,`(if` $=, 3$ `then 5, else` p_4`)`)
(5,`(move 1 right)`)
(6,`(if` $1 = \lambda$ `then 6, else 1)`)
(7,`(compute Id,`1 `to 2)`)
(8,`(move 2 left)`)
(9,`(compute` $+, 2$ `to 2)`)
(10,`(move 2 right)`)
(11,`(bin-guess 3)`)
(12,`(move 3 left)`)
(13,`(compute 0 to 3)`)
(14,`(if` $=, 3$ `then 1, else 15)`)
(15,`(compute 0 to 2)`)
(16,`(move 2 left)`)
(17,`(if` $=, 2$ `then 18, else 20)`)
(18,`(compute 1 to 3)`)
(19,`(stop)`)
(20,`(compute 0 to 3)`)
(21,`(stop)`)

Die Maschine wählt also mit Zeile 1 bis 4 unter Verwendung des dritten Bandes und der nichtdeterminstischen Rateanweisung nichtdeterministisch, ob sie die aktuelle Komponente der Eingabe verwendet (dann springt sie in Zeile 7), oder mit einem Sprung in Zeile 5 übergeht. Wird die Komponente verwendet, so kopiert die Maschine sie mit den Zeilen 7 bis 10 auf ihr Arbeitsband und addiert sie zu dem bisherigen Arbeitswert hinzu. Hierbei ist zu Beachten, daß nach Definition bei der Berechnung einer Funktion als Argument 0 eingesetzt wird, falls das entsprechende Feld leer ist. Nun springt sie mit den Zeilen 11

bis 14 nichtdeterministisch entweder wieder an den Anfang oder testet mit den Zeilen 15 bis 17, ob der Arbeitswert gerade null ist, und gibt dann mit den Zeilen 18 bis 21 das entsprechende Ergebnis aus.

Die Maschine $\mathbb{A}$ akzeptiert die „Ja"-Instanzen des Problems NULLSACK, denn falls es für eine Eingabe $z_1 \dots z_n$ eine Indexmenge $I \subseteq [n]$ mit $\sum_{i \in I} z_i = 0$ gibt, so gibt es eine entsprechende Berechnung von $\mathbb{A}$, die gerade die richtigen z_i addiert und 1 ausgibt, falls es jedoch für $z_1 \dots z_n$ keine derartige Indexmenge gibt, so gibt jede Berechnung von $\mathbb{A}$ angesetzt auf $z_1 \dots z_n$ entweder 0 oder gar nichts aus, weil sie irgendwann in eine Endlosschleife gerät.

Um die Komplexität von Problemen beschreiben zu können, ist es wichtig, einen ähnlichen Begriff wie den der Laufzeit auch für binär nichtdeterministische Maschinen zur Verfügung zu haben. Da in der obigen Definition nur wichtig ist, ob es eine ein Element akzeptierende Berechnung gibt, beschränken wir uns auch bei der Definition der benötigten Zeit auf diese eine akzeptierende Berechnung.

Definition 2.2.4 Sei T eine Funktion von $\mathbb{N}$ nach $\mathbb{R}$. Eine binär nichtdeterminische Turingmaschine $\mathbb{A}$ über der Struktur $\mathcal{M}$ *akzeptiert eine Sprache* $L \subseteq M^*$ *in Zeit* $T(n)$, falls $\mathbb{A}$ die Sprache L akzeptiert und es für jedes Element $\bar{x} \in L$ eine akzeptierende Berechnung von $\mathbb{A}$ gibt, welche höchstens $T(|\bar{x}|)$ Schritte lang ist.

Wenn eine binär nichtdeterministische Turingmaschine ein Problem in einer bestimmten Zeit akzeptiert, so ist für jede Eingabe die Anzahl der möglichen Berechnungen beschränkt, da in jedem Berechnungsschritt höchstens zwei Alternativen zur Auswahl stehen. Es ist also möglich, alle diese möglichen Berechnungen bis zu einem Zeitpunkt t mit einer deterministischen Turingmaschine zu simulieren. Diese Tatsache liefert das folgende nützliche Lemma.

Lemma 2.2.5 *Sei $X \subseteq M^*$ ein Problem über einer Struktur $\mathcal{M}$, welches von einer binär nichtdeterministischen Turingmaschine über $\mathcal{M}$ für ein Polynom $p(n)$ in Zeit $p(n)$ akzeptiert wird. Dann gibt es eine Konsante $C \in \mathbb{N}$, ein Polynom $q(n)$ und eine deterministische Turingmaschine über $\mathcal{M}$, welche X in Zeit $C \cdot p(n) \cdot 2^{C \cdot p(n)} + q(n)$ entscheidet.*

Beweis: Sei also $\mathbb{A}$ eine binär nichtdeterministische Turingmaschine über $\mathcal{M}$, die X in Zeit $p(n)$ akzeptiert. Sei $\mathbb{A}'$ nun eine deterministische Turingmaschine über $\mathcal{M}$, die bei Eingabe von $\bar{x}$ den Wert $p(|\bar{x}|)$ berechnet, die ersten $p(|\bar{x}|)$ Schritte jeder möglichen Berechnung von $\mathbb{A}$ simuliert, 1 ausgibt, falls eine dieser Berechnungen $\bar{x}$ akzeptiert, andernfalls 0. Es ist klar, daß $\mathbb{A}'$ genau dann 1 ausgibt, wenn $\mathbb{A}$ die Eingabe $\bar{x}$ akzeptiert. Die Berechnung des Wertes $p(|\bar{x}|)$ ist für ein Polynom $q(n)$ in $q(|\bar{x}|)$ Schritten möglich. Da in den ersten $p(|\bar{x}|)$ Schritten einer jeden möglichen Berechnung von $\mathbb{A}$ auch nur höchstens $p(|\bar{x}|)$ nichtdeterministische Sprunganweisungen vorkommen können, gibt es höchstens $2^{p(|\bar{x}|)}$ viele verschiedene Berechnungen von $\mathbb{A}$, die die deterministische Maschine $\mathbb{A}$ simulieren muß. Somit läuft $\mathbb{A}$ bei Eingabe von $\bar{x}$ für eine geeignete Konstante C also höchstens $C \cdot p(|\bar{x}|) \cdot 2^{C \cdot p(|\bar{x}|)} + q(|\bar{x}|)$ viele Schritte. □

Nach diesem Lemma ermöglichen binär nichtdeterministische Maschinen gegenüber deterministischen Maschinen also nicht grundsätzlich neue Berechnungen. Allerdings könnte es sein, daß es eine binär nichtdeterministische Turingmaschine gibt, so daß jede der diese Maschine simulierenden deterministischen Turingmaschinen einen exponentiellen Zeitverlust hinnehmen muß.

Bemerkung 2.2.6 Eine binär nichtdeterministische Turingmaschine über der Standardstruktur entspricht einer nichtdeterministischen Turingmaschine im klassischen Sinne. Für eine Beschreibung des klassischen Falles siehe zum Beispiel [21] oder [12].

2.2.2 Die Klassen $\mathrm{BNP}_{\mathcal{M}}$ und $\mathrm{BVP}_{\mathcal{M}}$

Analog zum deterministischen Fall läßt sich auch im binär nichtdeterministischen Fall eine Klasse von „handhabbaren“ Problemen mittels der Laufzeit einer ein Problem entscheidenden binär nichtdeterministischen Turingmaschine definieren. Wir werden sehen, daß die so definierte Klasse $\mathrm{BNP}_{\mathcal{M}}$ mit der in [22] oder [3] behandelten Klasse $\mathrm{BVP}_{\mathcal{M}}$, welche über die schnelle Verifizierbarkeit bei Vorlage eines binären Zeugens definiert ist, übereinstimmt.

Definition 2.2.7 ($\mathrm{BNP}_{\mathcal{M}}$) Sei $\mathcal{M}$ eine Struktur. Ein Problem $X \subseteq M^*$ ist in der Klasse $\mathrm{BNP}_{\mathcal{M}}$, falls es eine binär nichtdeterministische Turingmaschine über der Struktur $\mathcal{M}$ gibt, welche X in polynomieller Zeit akzeptiert.

Aus Lemma 2.2.5 folgt dann sofort, daß es für jedes Problem $X \in \mathrm{BNP}_{\mathcal{M}}$ eine deterministische Maschine gibt, die X in exponentieller Zeit entscheidet:

Korollar 2.2.8 *Für jede Struktur $\mathcal{M}$ gilt:* $\mathrm{BNP}_{\mathcal{M}} \subseteq \mathrm{EXP}_{\mathcal{M}}$.

□

Betrachten wir die Klasse $\mathrm{BVP}_{\mathcal{M}}$:

Definition 2.2.9 ($\mathrm{BVP}_{\mathcal{M}}$) Sei $\mathcal{M}$ eine Struktur. Ein Problem $X \subseteq M^*$ ist in $\mathrm{BVP}_{\mathcal{M}}$ genau dann, wenn es ein Problem $Y \in \mathrm{P}_{\mathcal{M}}$ und ein Polynom q gibt, so daß für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \text{es gibt ein } \bar{\varepsilon} \in \{0,1\}^{q(|\bar{x}|)} \text{ mit } \bar{x}\bar{\varepsilon} \in Y$$

Wie hängt nun diese Klasse mit der oben betrachteten Klasse $\mathrm{BNP}_{\mathcal{M}}$ zusammen?

Die exponentielle Verlangsamung in Lemma 2.2.5 kam dadurch zustande, daß die deterministische Maschine *alle* möglichen Berechnungen der binär nichtdeterministischen Maschine bis zu einem bestimmten Zeitpunkt simulieren mußte. Wenn die deterministische Maschine nun allerdings eine Information darüber hat, welche der möglichen Berechnungen der binär nichtdeterministischen Maschine sie simulieren soll, so kann sie dies ohne nennenswerten Zeitverlust tun und das Ergebnis der entsprechenden Berechnung der nichtdeterministischen Maschine ausgeben. Wenn also eine binär nichtdeterministische Maschine ein Problem X in polynomieller Zeit akzeptiert, kann eine deterministische Maschine bei gegebenem $\bar{x} \in M^*$ und Angabe zur Berechnung der nichtdeterministischen

Maschine in polynomieller Zeit überprüfen, ob die entsprechende Berechnung der binär nichtdeterministischen Maschine die Eingabe $\bar{x}$ akzeptiert. Falls $\bar{x} \in X$, so gibt es eine Angabe zu einer akzeptierenden Berechnung, andernfalls nicht. Eine Angabe zu einer Berechnung von einer binär nichtdeterministischen Maschine besteht dabei in einer Angabe der jeweils geratenen Elemente. Da die Maschine nur (polynomiell viele) Elemente aus $\{0, 1\}$ raten kann, reicht eine 0-1 Folge polynomieller Länge zur Angabe einer Berechnung aus. In diesem Sinne können wir die Gleichheit von $\mathrm{BNP}_{\mathcal{M}}$ und $\mathrm{BVP}_{\mathcal{M}}$ beweisen:

Satz 2.2.10 ($\mathrm{BNP}_{\mathcal{M}} = \mathrm{BVP}_{\mathcal{M}}$) *Für jede Struktur $\mathcal{M}$ gilt*

$$\mathrm{BNP}_{\mathcal{M}} = \mathrm{BVP}_{\mathcal{M}}$$

Beweis: „$\subseteq$“: Sei also $X \in \mathrm{BNP}_{\mathcal{M}}$, und sei $\mathbb{A}$ eine binär nichtdeterministische Turingmaschine über $\mathcal{M}$, welche das Problem X in polynomieller Zeit akzeptiert. Sei weiterhin k die Anzahl der Bänder von $\mathbb{A}$ und $q(n)$ das Polynom, welches die Länge der akzeptierenden Berechnungen von $\mathbb{A}$ beschränkt. Ohne Einschränkung ist q monoton wachsend. Wenn ein $\bar{x} \in M^*$ akzeptiert wird, so gibt es eine akzeptierende Berechnung von $\mathbb{A}$, die höchstens $q(|\bar{x}|)$ Schritte lang ist. In einer solchen Berechnung kommen also auch nur höchstens $q(|\bar{x}|)$ nichtdeterministische Rateanweisungen vor.

Wir konstruieren eine deterministische Turingmaschine $\mathbb{A}'$ über $\mathcal{M}$ mit $k+1$ Bändern wie folgt: Bei Eingabe von $\bar{z} \in M^*$ berechnet $\mathbb{A}'$ zunächst mit den Elementen 0 und 1 und mit der Selektorfunktion S wie eine klassische Turingmaschine den Wert $|\bar{z}|$. Daraufhin berechnet sie, ob es eine natürliche Zahl n gibt mit $|\bar{z}| = n + q(n)$. Dazu testet sie einfach für alle $i \in [|\bar{z}|]$, ob $i + q(i) = |\bar{z}|$. Gibt es ein solches n, so prüft die Maschine, ob $z_{n+1}, \ldots, z_{n+q(n)} \in \{0, 1\}$ gilt, und hält an und gibt 0 aus, falls nicht. Andernfalls fährt sie in der Berechnung fort. Wenn es eine solche Zahl n nicht gibt, hält die Maschine ebenso an und gibt 0 aus. Dies stellt sicher, daß die Eingabe die richtige Form hat.

Gibt es ein solches n, so hat die Eingabe also die Form $x_1 \ldots x_n y_1 \ldots y_{q(n)}$. Die Maschine kopiert nun $y_1, \ldots, y_{q(n)}$ auf das k-te Band und setzt die verwendeten Zeiger wieder zurück auf ihre Ausgangsposition. Das k-te Band dient in Folge nur noch dazu, die einzelnen y_i auszulesen. Daraufhin läuft eine modifizierte Version des Programmes der Maschine $\mathbb{A}$ ab. Das Programm wird dabei in einem ersten Schritt dahingehend modifiziert, daß jeder Verweis auf das Ausgabeband, welches ja das k-te Band der Maschine $\mathbb{A}$ war, in einen Verweis auf das neue Ausgabeband, das $k+1$-te Band, umgeändert wird. Dieser Schritt ist notwendig, da ja nach Konvention das letzte Band als Ausgabeband genutzt wird.

In einem zweiten Schritt wird vor jeder Anweisung ein Test eingebaut, ob das durch den Zeiger des k-ten Bandes markierte Feld leer ist. Sollte dies der Fall sein, so bricht die Maschine die Berechnung ab und gibt 0 aus. Ist das Feld nicht leer, so führt sie die entsprechende Anweisung aus und bewegt danach den Zeiger des k-ten Bandes um eine Position nach rechts. Diese Modifikation garantiert, daß bei einer Eingabe $\bar{x}\bar{y}$ höchstens $q(\bar{x})$ Schritte des Programmes von $\mathbb{A}$ durchlaufen werden. Außerdem wird auf diese Weise jedes y_i nur einmal benutzt.

Im dritten und letzten Schritt schließlich wird jede nichtdeterministische Rateanweisung (`bin-guess` i) durch die Anweisung (`compute Id,`k `to` i) ersetzt. Hierdurch wird die nichtdeterministische Komponente des Programmes entfernt.

Bei Eingabe von $\bar{z} = x_1 \dots x_n y_1 \dots y_{q(n)}$ simuliert die auf diese Weise konstruierte Maschine $\mathbb{A}'$ also die Berechnung von $\mathbb{A}$ bei Eingabe von $\bar{x}$, welche durch $\bar{y}$ kodiert wird. Sei also $Y := \{\bar{x}\bar{y} \mid \mathbb{A}' \text{ akzeptiert } \bar{x}\bar{y}\}$. Das Problem Y ist dann in der Klasse $\mathrm{P}_{\mathcal{M}}$, da die Maschine $\mathbb{A}'$ nur polynomiell viele Schritte läuft:

Die Überprüfung, ob die Eingabe die richtige Form hat, erfordert nur polynomiell viele Schritte, da die benötigte Zeit um einen Wert $i + q(i)$ für $i < |\bar{z}|$ zu berechnen höchstens so groß ist, wie die zur Berechnung des Wertes $|\bar{z}| + q(|\bar{z}|)$ benötigte Zeit (da q ohne Einschränkung monoton); da der Wert $|\bar{z}| + q(|\bar{z}|)$ in polynomieller Zeit im klassischen Sinne berechnet werden kann; und da jeder der berechneten Werte in polynomieller Zeit mit der Länge der Eingabe verglichen werden kann. Das Kopieren von $y_1, \dots, y_{q(n)}$ und das Rücksetzen der Zeiger können offensichtlicherweise in polynomieller Zeit bewerkstelligt werden, und der zusätzlich eingebaute Test verlängert die Laufzeit des Programmes nur um einen linearen Faktor, da jedesmal die gleiche Prozedur durchlaufen wird. Das Ersetzen der nichtdeterministischen Rateanweisungen verlängert die Laufzeit überhaupt nicht.

Für das so definierte Y gilt dann: $\bar{x} \in X \Leftrightarrow$ es gibt ein $\bar{\varepsilon} \in \{0,1\}^{q(|\bar{x}|)}$ mit $\bar{x}\bar{\varepsilon} \in Y$, denn:

Für ein $\bar{x} \in X$ gibt es eine akzeptierende Berechnung von $\mathbb{A}$ und somit ein Wort $\varepsilon_1 \dots \varepsilon_{q(|\bar{x}|)}$ aus $\{0,1\}^*$, welches die $\bar{x}$ akzeptierende Berechnung von $\mathbb{A}$ kodiert. Für das Wort $\bar{\varepsilon}$ gilt somit $\bar{x}\bar{\varepsilon} \in Y$.

Wenn es andererseits ein $\bar{\varepsilon} \in \{0,1\}^{q(|\bar{x}|)}$ gibt, so daß $\bar{x}\bar{\varepsilon} \in Y$ gilt, dann kodiert dieses $\bar{\varepsilon}$ nach Konstruktion von $\mathbb{A}'$ eine akzeptierende Berechnung von $\mathbb{A}$ bei Eingabe von $\bar{x}$. Somit gilt dann $\bar{x} \in X$.

Also gilt $X \in \mathrm{BVP}_{\mathcal{M}}$.

„$\supseteq$“: Sei andersherum $X \in \mathrm{BVP}_{\mathcal{M}}$. Dann gibt es ein Problem $Y \in \mathrm{P}_{\mathcal{M}}$ und ein Polynom $q(n)$, so daß für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \text{es gibt ein } \bar{\varepsilon} \in \{0,1\}^{q(|\bar{x}|)} \text{ mit } \bar{x}\bar{\varepsilon} \in Y.$$

Sei weiterhin $\mathbb{A}$ die das Problem Y in Zeit $p(n)$ entscheidende deterministische Turingmaschine über $\mathcal{M}$. Eine binär nichtdeterministische Turingmaschine über $\mathcal{M}$, welche X akzeptiert, arbeitet dann wie folgt:

Zunächst berechnet sie wie eine klassische Turingmaschine in polynomieller Zeit den Wert $q(|\bar{x}|)$. Dann rät sie mittels dieses berechneten Wertes und der nichtdeterministischen Rateanweisung ein Wort $\bar{\varepsilon} \in \{0,1\}^{q(|\bar{x}|)}$. Auch dieses gelingt in polynomieller Zeit. Abschließend simuliert sie in $\mathcal{O}(p(q(n)+n))$ Schritten die Maschine $\mathbb{A}$ mit Eingabe $\bar{x}\bar{\varepsilon}$ und gibt die entsprechende Ausgabe von $\mathbb{A}$ aus. Nach Konstruktion akzeptiert diese Maschine also das Problem X in polynomieller Zeit. Somit gilt $X \in \mathrm{BNP}_{\mathcal{M}}$. □

Die Gleichheit der Klassen $\mathrm{BNP}_{\mathcal{M}}$ und $\mathrm{BVP}_{\mathcal{M}}$ erlaubt manchmal elegantere Beweise, da man nicht mehr unbedingt einen binär nichtdeterministischen Algorithmus angeben muß:

Beispiel 3 (KNAPSACK) Sei $\mathcal{M} = (\mathbb{N}, +, \cdot)$. Wir betrachten das Problem KNAPSACK, ein ähnliches Problem, wie das in Beispiel 2 behandelte Problem NULLSACK:

KNAPSACK
Eingabe: $\bar{x} = x_1 \dots x_n \in \mathbb{N}^*$
Frage: Gibt es ein $I_{\bar{x}} \subseteq \{2, 3, \dots, n\}$ mit $x_1 = \sum_{i \in I_{\bar{x}}} x_i$?

Das Problem KNAPSACK ist also das Problem, bei Eingabe von endlich vielen natürlichen Zahlen $x_1, \dots, x_n$ zu entscheiden, ob x_1 durch Addition von Zahlen aus der Menge $\{x_2, \dots, x_n\}$ gewonnen werden kann. Die „Ja"-Instanzen von KNAPSACK sind also für jedes $n \in \mathbb{N}$ alle Wörter $\bar{x} = x_1 \dots x_n$, für die es ein $I_{\bar{x}} \subseteq \{2, 3, \dots, n\}$ gibt, so daß gilt: $x_1 = \sum_{i \in I_{\bar{x}}} x_i$, was gleichbedeutend ist dazu, daß es ein Wort $\bar{\varepsilon} \in \{0,1\}^{n-1}$ mit $x_1 = \sum_{k=2}^{n} \varepsilon_{i-1} \cdot x_i$ gibt. Da ein Wert $\sum_{k=2}^{n} \varepsilon_{i-1} \cdot x_i$ über $\mathcal{M}$ in polynomieller Zeit berechnet und mit x_1 verglichen werden kann, ist KNAPSACK in $\mathrm{BVP}_{\mathcal{M}}$ und mit Satz 2.2.10 somit in der Klasse $\mathrm{BNP}_{\mathcal{M}}$.

2.2.3 Vollständig nichtdeterministische Turingmaschinen

Nichtdeterministische Turingmaschinen des zweiten Typs oder vollständig nichtdeterministische Turingmaschinen können nun nicht nur Elemente aus $\{0,1\}$, sondern Elemente aus dem Universum der jeweiligen Struktur raten. Wenn im folgenden von nichtdeterministischen Turingmaschinen die Rede ist, werden immer vollständig nichtdeterministische Turingmaschinen gemeint sein.

Definition 2.2.11 (vollständig nichtdeterministische Turingmaschine) Eine *vollständig nichtdeterministische Turingmaschine über* $\mathcal{M}$ ist eine deterministische Turingmaschine über $\mathcal{M}$, in deren Programm zusätzlich zu den deterministischen Befehlen

- `(move` i `right)` oder `(move` i `left)`
- `(compute` f`,`i `to` j`)`
- `(if` R`,`i `then` p`, else` q`)`
- `(if` $i = \lambda$ `then` p`, else` q`)`
- `(stop)`

mit den gewohnten Bedeutungen auch noch Rateanweisungen der Form

- `(guess` i`)`

vorkommen können. Wenn an einer Stelle in einer Berechnung eine Rateanweisung auftaucht, so rät die Maschine ein Element aus M und schreibt es in das durch den Zeiger des i-ten Bandes markierte Feld. Hierbei ist natürlich $i \in [k]$ die Nummer eines der Bänder der Maschine.

Die Begriffe der Konfigurationen, einer akzeptierenden Berechnung und einer eine Sprache akzeptierenden nichteterministischen Turingmaschine werden analog wie im binär nichtdeterministischen Fall definiert:

Definition 2.2.12 (akzeptiert Sprache L) Eine ein Element $\bar{x} \in M^*$ *akzeptierende Berechnung* einer nichdeterministischen Turingmaschine $\mathbb{A}$ über $\mathcal{M}$ ist eine Berechnung von $\mathbb{A}$ mit der $\bar{x}$ als Eingabe festlegenden Startkonfiguration und einer das Element 1 als Ausgabe festlegenden Haltekonfiguration. Die nichtdeterministische Maschine $\mathbb{A}$ *akzeptiert eine Sprache* $L \subseteq M^*$, falls es für jedes $\bar{x} \in L$ mindestens eine und für jedes $\bar{y} \notin L$ keine akzeptierende Berechnung von $\mathbb{A}$ gibt. Für eine Funktion $T : \mathbb{N} \to \mathbb{R}$ akzeptiert $\mathbb{A}$ die Sprache L *in Zeit* $T(n)$, falls L von $\mathbb{A}$ akzeptiert wird und es für jedes $\bar{x} \in L$ eine akzeptierende Berechnung von $\mathbb{A}$ mit höchstens $T(|\bar{x}|)$ Schritten gibt.

Beispiel 4 Ein natürliches Problem, welches mit Hilfe von nichtdeterministischen Turingmaschinen über einem Körper $\mathcal{K} = (K, +, -, \cdot)$ gelöst werden kann, ist das Problem $\mathrm{HNS}_{\mathcal{K}}$. Dies ist das Problem, bei Eingabe von n Polynomen über $\mathcal{K}$ in m Variablen zu entscheiden, ob die Polynome eine gemeinsame Nullstelle haben:

$\mathrm{HNS}_{\mathcal{K}}$ **Eingabe:** Polynome $p_1, \ldots, p_m \in K[X_1, \ldots, X_n]$ in Kodierung durch ihre Koeffizienten **Frage:** Gibt es ein $\bar{a} \in K^n$ mit $p_1(\bar{a}) = p_2(\bar{a}) = \ldots = p_m(\bar{a}) = 0$?

Eine dieses Problem lösende nichtdeterministische Turingmaschine über $\mathcal{K}$ rät nun bei Eingabe von $p_1, \ldots, p_m \in K[X_1, \ldots, X_n]$ einfach ein Wort $\bar{a}$ aus K^n und rechnet aus, ob $p_i(\bar{a}) = 0$ für alle $i \in [m]$ gilt. Ist dies der Fall, so gibt die Maschine 1 aus, andernfalls 0.

Bemerkung 2.2.13 Jede binär nichtdeterministische Turingmaschine über einer Struktur $\mathcal{M}$ läßt sich mit konstantem Zeitverlust durch eine vollständig nichtdeterministische Turingmaschine über $\mathcal{M}$ simulieren. Hierzu rät die vollständig nichtdeterministische Maschine für jeden binären Ratebefehl der binär nichtdeterministischen Maschine ein Element aus M und schreibt eine 0 auf das entsprechende Feld, falls das geratene Element die 0 war, andernfalls eine 1.

2.2.4 Die Klassen $\mathrm{NP}_{\mathcal{M}}$ und $\mathrm{VP}_{\mathcal{M}}$

Analog zu der über binär nichtdeterministische Turingmaschinen definierten Klasse $\mathrm{BNP}_{\mathcal{M}}$ läßt sich nun die über vollständig nichtdeterministische Turingmaschinen definierte Klasse $\mathrm{NP}_{\mathcal{M}}$ einführen, welche, wie wir sehen werden, mit der über schnelle Verifizierbarkeit einer "Ja"-Instanz zusammen mit einem Zeugen polynomieller Länge definierten Klasse $\mathrm{VP}_{\mathcal{M}}$ übereinstimmt.

Definition 2.2.14 ($\mathrm{NP}_{\mathcal{M}}$) Sei $\mathcal{M}$ eine Struktur mit Universum M. Ein Problem $X \subseteq M^*$ ist in der Klasse $\mathrm{NP}_{\mathcal{M}}$, falls es eine vollständig nichtdeterministische Turingmaschine über $\mathcal{M}$ gibt, welche X in polynomieller Zeit akzeptiert.

Der in [22] oder in [3] behandelte Begriff von Nichtdeterminismus weicht wie im binär nichtdeterministischen Fall auch hier von unserem über nichtdeterministische Turingmaschinen definierten Begriff von Nichtdeterminismus ab. Dort wird das Kriterium der schnellen Verifizierbarkeit einer "Ja"-Instanz bei gegebenem Zeugen verwendet. Dies fassen wir in folgende Definition:

Definition 2.2.15 ($\mathrm{VP}_{\mathcal{M}}$) Sei $\mathcal{M}$ eine Struktur mit Universum M. Ein Problem $X \subseteq M^*$ ist in der Klasse $\mathrm{VP}_{\mathcal{M}}$, falls es ein Problem $Y \in \mathrm{P}_{\mathcal{M}}$ und ein Polynom q gibt, so daß für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \text{ es gibt ein } \bar{y} \in M^{q(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in Y$$

Wenn es also für ein Wort $\bar{x}$ eine akzeptierende Berechnung einer nichtdeterministischen Turingmaschine $\mathbb{A}$ gibt, dann rät $\mathbb{A}$ also ein Wort $\bar{y} \in M^*$ und verifiziert mithilfe von $\bar{y}$ die Zugehörigkeit von $\bar{x}$ zu dem Problem X. In Analogie zu Satz 2.2.10 bedeutet das:

Satz 2.2.16 ($\mathrm{NP}_{\mathcal{M}} = \mathrm{VP}_{\mathcal{M}}$) *Für jede Struktur $\mathcal{M}$ gilt:*

$$\mathrm{NP}_{\mathcal{M}} = \mathrm{VP}_{\mathcal{M}}$$

Beweis: "$\subseteq$": Sei also $X \in \mathrm{NP}_{\mathcal{M}}$, und sei weiterhin $\mathbb{A}$ die X in Zeit p entscheidende nichtdeterministische Turingmaschine mit k Bändern. Im Programm von $\mathbb{A}$ können also Ratebefehle vorkommen.

Analog zum Beweis von Satz 2.2.10 konstruieren wir nun eine deterministische Turingmaschine $\mathbb{A}'$ mit $k+1$ Bändern, welche zunächst sicherstellt, daß die Eingabe die Form $x_1 \ldots x_n y_1 \ldots y_{q(n)}$ hat, dann die $y_1, \ldots, y_{q(n)}$ auf das k-te Band kopiert und schließlich das modifizierte Programm von $\mathbb{A}$ ausführt. Das Programm modifizieren wir analog zum binär nichtdeterministischen Fall, indem wir zunächst alle Verweise auf das k-te Band in Verweise auf das $k+1$-te Band umändern, vor jeder Anweisung den Test einbauen, ob das aktuelle Feld des k-ten Bandes leer ist, und nach jeder Anweisung den Befehl, den Zeiger des k-ten Bandes um eine Position nach rechts zu verschieben. Die Ratebefehle `(guess` i`)` werden analog zum binär nichtdeterministischen Fall durch den Befehl `(compute Id,` k `to` i`)` ersetzt, welcher den Inhalt des durch den Zeiger des k-ten Bandes markierten Feldes in das durch den Zeiger des i-ten Bandes markierte Feld kopiert.

Das Problem Y setzen wir wieder als die Menge der $\bar{x}\bar{y} \in M^*$, welche von der so konstruierten Maschine $\mathbb{A}'$ akzeptiert werden. Wie oben gilt dann $Y \in \mathrm{P}_{\mathcal{M}}$ und für alle $\bar{x} \in M^*$ die geforderte Aussage

$$\bar{x} \in X \Leftrightarrow \text{ es gibt ein } \bar{y} \in M^{q(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in Y.$$

„$\supseteq$": Wenn andererseits X aus $\mathrm{VP}_{\mathcal{M}}$ ist und es somit ein entsprechendes Problem $Y \in \mathrm{P}_{\mathcal{M}}$ und ein Polynom $q(n)$ gibt, so daß für alle $\bar{x} \in M^*$ gilt

$$\bar{x} \in X \Leftrightarrow \text{ es gibt ein } \bar{y} \in M^{q(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in Y,$$

so läßt sich auf folgende Weise eine nichtdeterministische Turingmaschine definieren, welche das Problem X akzeptiert: Sei $\mathbb{A}$ die Y in Zeit $q'(n)$ entscheidende deterministische Turingmaschine. Die nichtdeterministische Turingmaschine $\mathbb{A}'$ berechnet bei Eingabe $\bar{x} \in M^*$ zunächst wie eine klassische Turingmaschine den Wert $q(|\bar{x}|)$. Dann bewegt sie den Zeiger des ersten Bandes auf das erste leere Feld des ersten Bandes, führt mit Hilfe des errechneten Wertes $q(|\bar{x}|)$-mal die Anweisungen `(guess 1)` und `(move 1 right)` aus und

bewegt den Zeiger des ersten Bandes zurück auf das x_1 beinhaltende Feld. Somit beinhalten die Felder des Eingabebandes nun das Element $\bar{x}\bar{y}$ mit einem nichtdeterministischen $\bar{y} \in M^{q(|\bar{x}|)}$. Abschließend simuliert $\mathbb{A}'$ die Maschine $\mathbb{A}$ mit Eingabe $\bar{x}\bar{y}$.

Da das Berechnen von $q(|\bar{x}|)$ und das Raten der $q(|\bar{x}|)$ Elemente aus M jeweils in polynomieller Zeit zu bewerkstelligen sind, und da die Simulation von $\mathbb{A}$ nach Voraussetzung nur polynomielle Zeit (polynomiell in $|\bar{x}| + q(|\bar{x}|)$) benötigt, läuft die Maschine $\mathbb{A}$ insgesamt höchstens polynomiell viele Schritte.

Desweiteren gibt es nach Konstruktion für eine Eingabe $\bar{x} \in M^*$ genau dann eine akzeptierende Berechnung von $\mathbb{A}'$, wenn es ein $\bar{y} \in M^{q(|\bar{x}|)}$ mit $\bar{x}\bar{y} \in Y$ gibt. Die Maschine $\mathbb{A}'$ akzeptiert also das Problem X in polynomieller Zeit. Somit ist X in der Klasse $\mathrm{NP}_{\mathcal{M}}$. □

Da nach Korollar 2.1.14 die Inklusion $\mathrm{P}_{\mathcal{M}} \subseteq \mathbb{P}_{\mathcal{M}}$ gilt, folgt damit sofort:

Korollar 2.2.17 *Für jede Struktur $\mathcal{M}$ gilt*

$$\mathrm{NP}_{\mathcal{M}} \subseteq \mathbb{NP}_{\mathcal{M}}$$

□

Bemerkung 2.2.18 (NP=VP) Im klassischen Fall ergibt sich aus Satz 2.2.16 unmittelbar die Identität der Klassen NP und VP. Die Klasse VP ist gerade die Klasse $\mathrm{VP}_{\{0,1\}}$.

Anders als im binär nichtdeterministischen Fall muß allerdings nicht jedes Problem aus $\mathrm{NP}_{\mathcal{M}}$ auch in $\mathrm{EXP}_{\mathcal{M}}$ sein, denn für eine Struktur mit unendlichem Universum müsste ein „brute-force“-Algorithmus, der alle möglichen Zeugen nacheinander testet, einerseits unendlich viele Zeugen testen, würde also unter Umständen nicht halten. Andererseits müsste er alle möglichen Zeugen aus den Konstanten der Struktur konstruieren können, was zum Beispiel schon in der Struktur $(\mathbb{R}, +, -)$ nicht möglich ist. Dies schließt natürlich nicht aus, daß es vielleicht noch einen intelligenteren Algorithmus zur Lösung dieses Problems gibt. Falls zum Beispiel in einer Struktur mit unendlichem Universum die Gleichheit $\mathrm{P}_{\mathcal{M}} = \mathrm{NP}_{\mathcal{M}}$ gilt, so gilt natürlich auch die Inklusion $\mathrm{NP}_{\mathcal{M}} \subseteq \mathrm{EXP}_{\mathcal{M}}$. Die von M. Prunescu in [24] konstruierte Struktur ist ein Beispiel hierfür.

Kapitel 3

Turingmaschinen in Flußdiagrammdarstellung

Eine etwas andere Herangehensweise an den Begriff einer Berechnung über einer beliebigen Struktur bieten Flußdiagramme. Mit ihrer Hilfe lassen sich manche Algorithmen etwas lesbarer darstellen als mittels Angabe des entsprechenden Programmes. Im folgenden werden wir zunächst in Anlehnung an die in [2] und [3] von L. Blum, F. Cucker, M. Shub und S. Smale definierten Maschinen die Begriffe der deterministischen, binär nichtdeterministischen und vollständig nichtdeterministischen Turingmaschinen in Flußdiagrammdarstellung über einer Struktur $\mathcal{M}$ definieren. Abschließend werden wir zeigen, daß die so definierten Maschinen in Flußdiagrammdarstellung polynomiell äquivalent zu den entsprechenden im letzten Abschnitt definierten Maschinen sind, daß sich also deterministische Turingmaschinen durch deterministische Turingmaschinen in Flußdiagrammdarstellung mit einem höchstens polynomiellen Zeitverlust simulieren lassen und andersherum.

3.1 Deterministische Maschinen

Wie im vorherigen Abschnitt betrachten wir zunächst den deterministischen Fall. Da wir Berechnungen mit Eingaben beliebiger Länge durchführen wollen, benötigen wir zunächst noch eine Definition.

Definition 3.1.1 (M_∞) Für eine Menge M ist

$M_\infty := \{(\ldots, x_{-1}, x_0.x_1, x_2, x_3, \ldots) \mid x_i \in M \text{ für alle } i \in \mathbb{Z}, x_i = 0 \text{ für } |i| \text{ hinreichend groß}\}$.

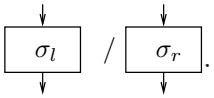
Für ein $(\ldots, x_{-1}, x_0.x_1, \ldots) \in M_\infty$ und eine ganze Zahl i heißt x_i die *i-te Komponente* von $(\ldots, x_{-1}, x_0.x_1, \ldots)$. Der Punkt zwischen der nullten und der ersten Komponente dient der besseren Lesbarkeit.

Die Elemente von M_∞ werden als Beschreibungen des Bandes unserer Maschinen dienen.

Definition 3.1.2 (TMFD/$\mathcal{M}$) Sei $\mathcal{M}$ eine Struktur. Eine *deterministische Turingmaschine in Flußdiagrammdarstellung über $\mathcal{M}$* ist ein endlicher zusammenhängender gerichteter Graph $\mathbb{F}$ mit gelabelten Knoten. Dabei gibt es fünf verschiedene Arten von Knoten:

- Es gibt genau einen *Eingangsknoten*. Dieser hat keine eingehenden und genau eine ausgehende Kante und ist mit $\mathcal{I}$ gelabelt:  . Hierbei ist $\mathcal{I}$ die Funktion

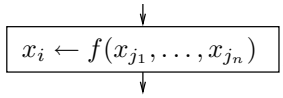
$$\mathcal{I} : M^* \to M_\infty, (x_1, \ldots, x_n) \mapsto (\ldots, 0, \overbrace{1, \ldots, 1}^{n\text{-mal}}.x_1, \ldots, x_n, 0, 0, \ldots).$$

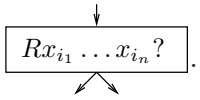
- *Shiftknoten* haben mindestens eine eingehende und genau eine ausgehende Kante und sind mit σ_l oder σ_r gelabelt:  .
  Hierbei sind σ_l und σ_r die Funktionen

$$\sigma_l : M_\infty \to M_\infty, (\sigma_l(x))_i = x_{i-1}$$

  beziehungsweise

$$\sigma_r : M_\infty \to M_\infty, (\sigma_r(x))_i = x_{i+1}.$$

- *Berechnungsknoten* haben mindestens eine eingehende und genau eine ausgehende Kante und sind mit $x_i \leftarrow f(x_{j_1}, \ldots, x_{j_n})$ für eine n-stellige Funktion f der Struktur $\mathcal{M}$ und $i, j_1, \ldots, j_n \in \mathbb{Z}$ gelabelt:  . Die Funktion f kann hierbei nullstellig, also eine Konstante, sein.

- *Verzweigungsknoten* haben mindestens eine eingehende und genau zwei ausgehende Kanten und sind mit $Rx_{i_1} \ldots x_{i_n}?$ für eine n-stellige Relation R der Struktur $\mathcal{M}$ und $i_1, \ldots, i_n \in \mathbb{Z}$ gelabelt:  .

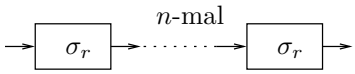
- Es gibt genau einen *Ausgangsknoten*, der mindestens eine eingehende und keine ausgehenden Kanten hat und mit $\mathcal{O}$ gelabelt ist:  . Hierbei ist $\mathcal{O}$ die Funktion

$$\mathcal{O} : M_\infty \to M^*, (\ldots, x_{-1}, x_0.x_1, \ldots) \mapsto (x_1, \ldots, x_{\min\{i \geq 0 | x_{-i} = 0\}}).$$

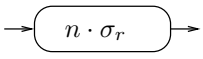
Falls ein Knoten η ein Verzweigungsknoten ist, dann sind die beiden Knoten, an denen die von η ausgehenden Kanten enden, zusätzlich mit $+$ beziehungsweise $-$ gelabelt. Wir bezeichnen den mit $+$ gelabelten Knoten dann mit $\beta^+(\eta)$ und den mit $-$ gelabelten Knoten mit $\beta^-(\eta)$. Ist η der Ausgangsknoten, so setzen wir $\beta(\eta) = \eta$. Ist η kein Verzweigungsknoten und kein Ausgangsknoten, so bezeichnen wir mit $\beta(\eta)$ den eindeutigen Knoten, an dem die von η ausgehende Kante endet. Mit η_{in} bezeichnen wir den eindeutigen Eingangsknoten, mit η_{out} den eindeutigen Ausgangsknoten. Mit $\mathcal{K}_\mathbb{F}$ bezeichnen wir die Menge der Knoten von $\mathbb{F}$.

Um eine kompaktere Darstellung zu ermöglichen, führen wir folgende Schreibweisen ein:

Schreibweise 3.1.3 Für eine Abfolge von n gleich gelabelten Knoten, beispielsweise den n Shiftknoten

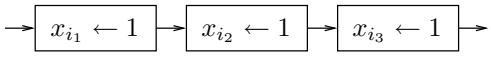


schreiben wir eine mit „$n \cdot \ldots$“ gelabelte Subroutine, im Beispiel die Subroutine

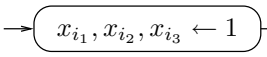


.

Falls in einer Abfolge von Knoten wie beispielsweise der Abfolge



mehrere Komponenten den gleichen Wert 1 zugeordnet bekommen, schreiben wir



.

Für den Fall, daß die Komponenten den gleichen Wert 0 zugeordnet bekommen analog.

Ein einfaches Beispiel einer deterministischen Turingmaschine in Flußdiagrammdarstellung über der Struktur $\mathcal{N} = (\mathbb{N}, +, =, 0, 1)$ findet sich in Abbildung 3.1. Die durch das Flußdiagramm dargestellte Turingmaschine testet, ob die erste Komponente der Eingabe größer als null und durch zwei teilbar ist. Ist dies der Fall, so hält sie an und gibt 1 aus, andernfalls hält sie nicht. Intuitiv ist eine Berechnung dieser Maschine bei Eingabe eines Wortes aus M^* ein Verfolgen des Flußdiagrammes, wobei die jeweiligen Anweisungen ausgeführt werden.

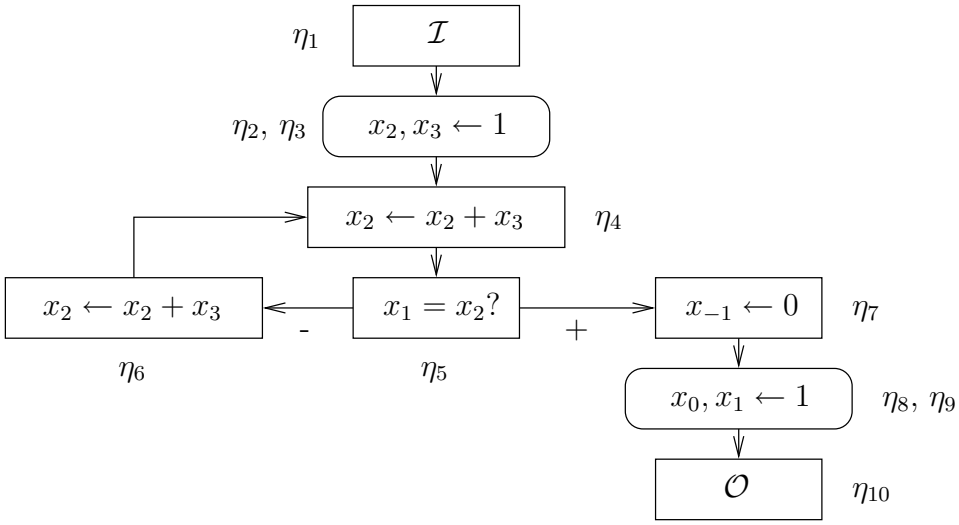


Abbildung 3.1: Eine einfache deterministische Turingmaschine in Flußdiagrammdarstellung über der Struktur $(\mathbb{N}, +, =, 0, 1)$

In unserem in Abbildung 3.1 dargestellten Beispiel wird für eine Eingabe $\bar{x} \in M^*$ zunächst im Eingangsknoten η_1 das Element $\mathcal{I}(\bar{x}) \in M_\infty$ berechnet. In den beiden in der Abbildung abgekürzt dargestellten Knoten η_2 und η_3 werden nun die zweite und dritte Komponente durch die Konstante 1 ersetzt. Anschließend wird zu der zweiten Kompo-

nente 1 addiert. Als nächstes wird im Verzweigungsknoten η_5 getestet, ob die erste Komponente gleich der zweiten Komponente ist, ob also die erste Komponente gleich zwei ist. Ist dies der Fall, so ist die erste Komponente durch zwei teilbar, und die Ausgabe wird mit den Knoten η_7 bis η_9 in die entsprechende Form gebracht und mit dem Ausgabeknoten η_{10} ausgegeben. Andernfalls wird in den Knoten η_6 und η_4 zu der zweiten Komponente noch zweimal 1 addiert, und der Test wiederholt.

Um den Begriff der Berechnung formal zu fassen, werden wir für eine deterministische Turingmaschine $\mathbb{F}$ in Flußdiagrammdarstellung über $\mathcal{M}$ für jede positive natürliche Zahl T eine Relation $\vdash_{\mathbb{F}}^{T} \subseteq M^* \times M^*$ definieren, welche auf zwei Elemente $\bar{x}$ und $\bar{y}$ aus M^* zutrifft, falls $\bar{y}$ eine mögliche Ausgabe von $\mathbb{F}$ bei Eingabe von $\bar{x}$ ist, wobei höchstens T Knoten durchlaufen werden. Hierzu benötigen wir zunächst noch einige technische Begriffe.

Ein Tupel $(\eta, w) \in \mathcal{K}_{\mathbb{F}} \times M_\infty$ lässt sich auffassen als Beschreibung eines Zustandes der Maschine $\mathbb{F}$. Hierbei beschreibt η den Knoten, in dem man sich gerade befindet, und w ist sozusagen der aktuelle Stand der Berechnung. Der entscheidende Schritt hierbei ist die folgende Definition der Relation $\vdash_{\mathbb{F}}$, welche auf zwei Zustände (η_1, v) und (η_2, w) zutrifft, falls es nach den durch $\mathbb{F}$ festgelegten Anweisungen möglich ist, vom Zustand (η_1, v) in den Zustand (η_2, w) überzugehen.

Definition 3.1.4 ($\vdash_{\mathbb{F}}$ für TMFD) Sei $\mathbb{F}$ eine deterministische Turingmaschine in Flußdiagrammdarstellung über einer Struktur $\mathcal{M}$. Die *Nachfolgerelation* $\vdash_{\mathbb{F}} \subseteq (\mathcal{K}_{\mathbb{F}} \times M_\infty) \times (\mathcal{K}_{\mathbb{F}} \times M_\infty)$ wird definiert durch:

Für $(\eta_1, v), (\eta_2, w) \in \mathcal{K}_{\mathbb{F}} \times M_\infty$ gilt $(\eta_1, v) \vdash_{\mathbb{F}} (\eta_2, w)$, falls einer der folgenden Fälle eintritt:

- η_1 ist mit $\mathcal{I}$ gelabelt und $\eta_2 = \beta(\eta_1)$ und $w = v$
- η_1 ist mit σ_l gelabelt und $\eta_2 = \beta(\eta_1)$ und $w = \sigma_l(v)$
- η_1 ist mit σ_r gelabelt und $\eta_2 = \beta(\eta_1)$ und $w = \sigma_r(v)$
- η_1 ist mit $x_i \leftarrow f(x_{j_i}, \ldots, x_{j_n})$ gelabelt und $\eta_2 = \beta(\eta_1)$ und
$$w_k = \begin{cases} f(v_{j_1}, \ldots, v_{j_n}), & \text{falls } k = i \\ v_k, & \text{sonst.} \end{cases}$$
- η_1 ist mit $Rx_{i_1}, \ldots, x_{i_n}?$ gelabelt und $w = v$ und
$$\eta_2 = \begin{cases} \beta^+(\eta_1), & \text{falls } \mathcal{M} \models Rv_{i_1}, \ldots, v_{i_n} \\ \beta^-(\eta_1), & \text{sonst.} \end{cases}$$
- η_1 ist mit $\mathcal{O}$ gelabelt und $\eta_2 = \beta(\eta_1) = \eta_1$ und $w = v$.

Mit der Relation $\vdash_{\mathbb{F}}$ lässt sich nun eine formale Definition der Zeit-T-Eingabe-Ausgabe-Relation $\vdash_{\mathbb{F}}^{T}$ angeben:

Definition 3.1.5 ($\vdash_{\mathbb{F}}^{T}$ für TMFD) Sei $\mathbb{F}$ eine deterministische Turingmaschine in Flußdiagrammdarstellung über einer Struktur $\mathcal{M}$. Die *Zeit-T-Eingabe-Ausgabe-Relation* $\vdash_{\mathbb{F}}^{T} \subseteq M^* \times M^*$ wird definiert durch:

Für $\bar{x}, \bar{y} \in M^*$ und $T \in \mathbb{N}_{>0}$ gilt $\bar{x} \vdash_{\mathbb{F}}^{T} \bar{y}$ genau dann, wenn $z^0, \ldots, z^T \in \mathcal{K}_{\mathbb{F}} \times M_\infty$ existieren, so daß

- $z^0 = (\eta_{in}, \mathcal{I}(\bar{x}))$
- $z^T = (\eta_{out}, z)$ für ein $z \in M_\infty$ mit $\mathcal{O}(z) = \bar{y}$
- für alle $i \in [T]$ gilt $z^{i-1} \vdash_{\mathbb{F}} z^i$.

Wenn $\bar{x} \vdash_{\mathbb{F}}^{T} \bar{y}$ für $\bar{x}, \bar{y} \in M^*$ und $T \in \mathbb{N}$ gilt, so gilt offensichtlicherweise auch $\bar{x} \vdash_{\mathbb{F}}^{T'} \bar{y}$ für alle $T' > T$.

Bemerkung 3.1.6 Die auf diese Weise definierten Turingmaschinen in Flußdiagrammdarstellung sind im wesentlichen Blum-Shub-Smale-Maschinen, wie sie in [2] beziehungsweise [3] definiert werden. Im Unterschied zu den hier definierten Maschinen können Blum-Shub-Smale-Maschinen allerdings in einem Knoten gleichzeitig mehrere Funktionen berechnen. Dies läßt sich mit Turingmaschinen in Flußdiagrammdarstellung mit konstantem Zeitverlust simulieren, indem die Funktionen einzeln in einem entsprechend großen Zwischenspeicher berechnet werden.

3.1.1 Der Zusammenhang zwischen deterministischen Turingmaschinen und deterministischen Turingmaschinen in Flußdiagrammdarstellung

Intuitiv ist es einleuchtend, daß deterministische Turingmaschinen und deterministische Turingmaschinen in Flußdiagrammdarstellung über einer Struktur dasselbe leisten. Es können ja jeweils dieselben Funktionen berechnet und dieselben Relationen überprüft werden. Der formale Beweis dieser intuitiven Tatsache gestaltet sich jedoch leider als etwas technisch, da wir zum einen der benötigten Zeit und zum anderen dem Umstand, daß Turingmaschinen mindestens zwei Bänder, Turingmaschinen in Flußdiagrammdarstellung jedoch nur ein Band haben, Rechnung tragen müssen. Wir zeigen zunächst, daß sich jede Turingmaschine in Flußdiagrammdarstellung durch eine Turingmaschine simulieren lässt.

Satz 3.1.7 („Für jede TMFD gibt es eine polynomiell äquvalente TM“) *Sei $\mathbb{F}$ eine deterministische Turingmaschine in Flußdiagrammdarstellung über einer Struktur $\mathcal{M}$. Dann gibt es eine deterministische Turingmaschine $\mathbb{A}_{\mathbb{F}}$ im Sinne von $\mathcal{M}$ und Poynome $p(n)$ und $q_{\mathbb{F}}(n)$, so daß für alle $\bar{x}, \bar{y} \in M^*$ und alle $T \in \mathbb{N}_{>0}$ gilt:*

$$\bar{x} \vdash_{\mathbb{F}}^{T} \bar{y} \quad \Leftrightarrow \quad \begin{array}{l} \mathbb{A}_{\mathbb{F}} \text{ hält bei Eingabe von } \bar{x} \text{ nach höchstens} \\ p(|\bar{x}|) + q_{\mathbb{F}}(T) \text{ Schritten und gibt } \bar{y} \text{ aus.} \end{array}$$

Beweis: Sei $\mathbb{F}$ eine deterministische Turingmaschine in Flußdiagrammdarstellung über $\mathcal{M}$. Eine deterministische Turingmaschine $\mathbb{A}_{\mathbb{F}}$ über $\mathcal{M}$, welche die Berechnung von $\mathbb{F}$ simuliert, wird wie folgt konstruiert: Zunächst berechnet $\mathbb{A}_{\mathbb{F}}$ bei Eingabe von $\bar{x}$ die relevanten Komponenten von $\mathcal{I}(\bar{x})$. Dies gelingt in $p(|\bar{x}|)$ Schritten. Dann führt $\mathbb{A}_{\mathbb{F}}$ nacheinander diejenigen Aktionen aus, die durch $\mathbb{F}$ festgelegt sind – die Maschine $\mathbb{A}_{\mathbb{F}}$ „arbeitet das Flußdiagramm ab“. Hierbei lässt sich erreichen, daß für jeden Knoten die selbe Anzahl Schritte benötigt wird. Zu guter Letzt wird die Ausgabe von $\mathbb{F}$ mittels der Outputabbildung $\mathcal{O}$ in die richtige Form gebracht und auf das Ausgabeband geschrieben. Dies wird

auf eine solche Weise erledigt, daß genau $q'(k)$ Schritte benötigt werden, falls vorher k Knoten durchlaufen wurden.

Formaler: Die Maschine $\mathbb{A}_{\mathbb{F}}$ hat fünf Bänder. Das erste Band wird ausschließlich als Eingabeband, das letzte Band als Ausgabeband genutzt. Auf dem zweiten Band werden die in der Berechnung von $\mathbb{F}$ verwendeten Elemente von M_∞ dargestellt, das dritte Band wird benutzt, um Berechnungen durchzuführen, und das vierte Band dient als Zähler für die durchlaufenen Knoten und die Länge der Eingabe. Das Programm von $\mathbb{A}$ wird in mehreren Schritten konstruiert. Im ersten Schritt stellen wir sicher, daß die Maschine $\mathbb{A}_{\mathbb{F}}$ genau das tut, was durch $\mathbb{F}$ gefordert ist. Danach müssen wir noch dafür sorgen, daß die benötigte Zeit die in der Aussage des Satzes geforderte Bedingung erfüllt. Für eine explizite Darstellung des Programmes siehe Anhang A.

1. Schritt: Zunächst wird für jeden Knoten $\eta \in \mathcal{K}_{\mathbb{F}}$ beginnend mit dem Eingangsknoten und endend mit dem Ausgangsknoten ein Block von Programmzeilen hinzugefügt, der den jeweiligen Knoten simuliert. Im Folgenden sei q_η die Nummer derjenigen Zeile, mit der der zu dem Knoten η gehörige Block anfängt. Desweiteren bezeichne $\bar{x}$ die Eingabe. Die einzelnen Blöcke haben dann die folgende Gestalt:

- Falls $\eta = \eta_{in}$, so wird die Länge von $\bar{x}$ berechnet und damit $\mathcal{I}(\bar{x})$ auf dem zweiten Band dargestellt. Die Tatsache, daß unendlich viele Komponenten von $\mathcal{I}(\bar{x})$ gleich null sind, ist hierbei unproblematisch, da nach der Definition der deterministischen Turingmaschine in dem Fall, daß eine Funktion oder Relation auf die Inhalte von leeren Feldern angewendet werden soll, für die Inhalte dieser Felder die Null eingesetzt wird. Die Maschine schreibt also $|\bar{x}|$-viele Einsen gefolgt von $\bar{x}$ auf das zweite Band und positioniert den Zeiger dieses Bandes über demjenigen Feld, welches die letzte Eins vor $\bar{x}$ enthält. Dann springt es in die Zeile $q_{\beta(\eta)}$.

- Falls η mit σ_l gelabelt ist, wird der Zeiger des zweiten Bandes um eine Position nach links verschoben. Anschließend springt das Programm in die Zeile $q_{\beta(\eta)}$.

- Falls η mit σ_r gelabelt ist, wird der Zeiger des zweiten Bandes um eine Position nach rechts verschoben. Anschließend springt das Programm in die Zeile $q_{\beta(\eta)}$.

- Falls η mit $x_i \leftarrow f(x_{j_1}, \ldots, x_{j_n})$ gelabelt ist, wird auf dem dritten Band der Wert von $f(x_{j_1}, \ldots, x_{j_n})$ berechnet und dann an die entsprechende Stelle des zweiten Bandes geschrieben. Der Zeiger des zweiten Bandes wird danach wieder an seine Ausgangsposition zurückverschoben. Anschließend springt das Programm in die Zeile $q_{\beta(\eta)}$.

- Falls η mit $Rx_{i_1} \ldots x_{i_n}?$ gelabelt ist, überprüft das Programm mit Hilfe des dritten Bandes, ob $Rx_{i_1} \ldots x_{i_n}$ gilt. Ist dies der Fall, so springt es in die Zeile $q_{\beta^+(\eta)}$, andernfalls in die Zeile $q_{\beta^-(\eta)}$.

- Falls $\eta = \eta_{out}$, so wird die entsprechende Ausgabe auf das Ausgabeband geschrieben. Hierzu berechnet die Maschine, in wie vielen Feldern beginnend mit und links von der Position des Zeigers des zweiten Bandes Einsen stehen, kopiert die Inhalte der entsprechenden Anzahl von Feldern rechts der Position des Zeigers des zweiten Bandes auf das Ausgabeband und setzt den Zeiger des Ausgabebandes zurück. Anschließend hält die Maschine an.

Nach Konstruktion gilt, daß die so konstruierte Maschine bei Eingabe von $\bar{x}$ genau dann hält und $\bar{y}$ ausgibt, wenn es eine natürliche Zahl t gibt, so daß $\bar{x} \vdash_{\mathbb{F}}^{t} \bar{y}$ gilt.

2. Schritt: Nun müssen wir noch dafür sorgen, daß die Maschine genau dann nach höchstens einer bestimmten Anzahl von Schritten hält, wenn $\Psi_{\mathbb{F}}^{T}(\bar{x}, \bar{y})$ gilt. Dies gelingt, indem wir das Programm derart modifizieren, daß wir Polynome $p(n)$ und $q_{\mathbb{F}}(n)$ angeben können, so daß für alle $\bar{x}, \bar{y} \in M^*$ gilt:

$$\bar{x} \vdash_{\mathbb{F}}^{\min\{t \mid \bar{x} \vdash_{\mathbb{F}}^{t} \bar{y}\}} \bar{y} \Leftrightarrow \mathbb{A}_{\mathbb{F}}(\bar{x}) = \bar{y} \text{ nach genau } p(|\bar{x}|) + q_{\mathbb{F}}(\min\{t \mid \bar{x} \vdash_{\mathbb{F}}^{t} \bar{y}\}) \text{ Schritten.}$$

Hierzu betrachten wir zunächst den den Eingangsknoten simulierenden Block von Programmzeilen. Um die Länge der Eingabe zu berechnen, gehen wir der Reihe nach jedes Feld des Eingabebandes durch, bis wir auf ein leeres Feld stoßen, und schreiben für jedes dieser Felder eine 1 auf ein Feld des vierten Bandes. Anschließend bewegen wir den Zeiger des Eingabebandes wieder zurück auf seine ursprüngliche Position und schreiben für jedes Element der Eingabe eine 1 auf ein Feld des zweiten Bandes. Als nächstes kopieren wir die Eingabe auf die nächsten Felder des zweiten Bandes und positionieren den Zeiger des zweiten Bandes über dem letzten eine 1 enthaltenden Feld vor der Eingabe, indem wir wiederum das Eingabeband als Angabe für die Länge der Eingabe benutzen. Da wir die einzelnen Teile durch Hinzufügen von „Springe in die nächste Zeile“-Anweisungen auf die gleiche Länge bringen können, gibt es eine Konstante A, so daß hierfür bei Eingabe von $\bar{x}$ genau $4 \cdot A \cdot |\bar{x}|$ Schritte benötigt werden.

Betrachten wir die Knoten, welche nicht Eingangs- oder Ausgangsknoten sind. Die Länge der zugehörigen Blöcke ist nur von dem Label des jeweiligen Knotens abhängig. Wenn zum Beispiel ein Knoten mit $x_i \leftarrow f(x_{j_1}, \ldots, x_{j_n})$ gelabelt ist, so wird zunächst der Zeiger des zweiten Bandes um j_1 Felder nach rechts (beziehungsweise links, falls j_1 negativ ist) bewegt, der entsprechende Feldinhalt auf das Berechnungsband kopiert, der Zeiger des Berechnungsbandes um eine Position nach rechts und der Zeiger des zweiten Bandes um j_1 Felder nach links (beziehungsweise rechts) bewegt. Ebenso werden die Inhalte des j_2-ten bis j_n-ten Feldes auf das Berechnungsband kopiert. Dann wird der Zeiger des Berechnungsbandes um n Felder nach links bewegt und mithilfe des Befehls `(compute` $f, 3$ `to 3)` der Funktionswert berechnet. Abschließend wird der Zeiger des zweiten Bandes um i Felder nach rechts bewegt, der errechnete Wert in das entsprechende Feld kopiert, der Zeiger des zweiten Bandes wieder um i Felder nach links bewegt und die Sprunganweisung in die Anfangszeile des zum nächsten Knoten gehörigen Blockes ausgeführt.

Bezeichnen wir mit $C'_{\mathbb{F}}$ die maximale Anzahl von Schritten, die bei Durchlaufen eines dieser Blöcke benötigt wird. Durch Hinzufügen von „Springe in die nächste Zeile“-Anweisungen lassen sich wiederum alle diese Blöcke derart modifizieren, daß beim Durchlaufen jedes Blockes genau $C'_{\mathbb{F}}$ Schritte benötigt werden. Unsere Maschine simuliert also bei Eingabe von $\bar{x}$ in genau $4 \cdot A \cdot |\bar{x}| + C'_{\mathbb{F}} \cdot k$ Schritten den Eingangsknoten und die nächsten k Knoten von $\mathbb{F}$ bei Eingabe von $\bar{x}$, falls der Ausgabeknoten nicht einer der k Knoten ist.

Die Simulation des Ausgabeknotens erfordert etwas Vorsicht. Da die Länge der Ausgabe nur von der Länge der Eingabe, nur von der Anzahl der durchlaufenen Knoten oder von beiden Größen abhängen kann, könnte es bei einer „straight-forward“-Simulation der Outputabbildung passieren, daß unsere Maschine in der erlaubten Anzahl Schritte einen

Knoten zu viel simulieren kann. Dies lässt sich jedoch verhindern, indem wir die Maschine zählen lassen, wie lang die Eingabe ist und wieviele Blöcke durchlaufen wurden, und bei der Simulation der Outputabbildung für k durchlaufene Knoten genau $|\bar{x}| + k$ Felder betrachten, so daß die Simulation der Outputabbildung genau $p'(|\bar{x}|) + q'(k)$ Schritte benötigt.

Da das vierte Band schon bei der Simulation des Eingabeknotens benutzt wurde, um die Länge der Eingabe zu speichern, genügt es nun, bei der Simulation der anderen Knoten jeweils auf das nächste Feld des vierten Bandes eine 1 zu schreiben. Hierfür sind jeweils zwei Schritte nötig. Somit werden bei Durchlaufen jedes dieser Blöcke nun $C_{\mathbb{F}} = C'_{\mathbb{F}} + 2$ Schritte benötigt.

Um den Ausgabeknoten zu simulieren wird schließlich für jedes nichtleere Feld des vierten Bandes getestet, ob das sich unter dem Zeiger des zweiten Bandes befindende Feld eine 1 enthält. Ist dies der Fall, so werden die Zeiger des zweiten und vierten Bandes um jeweils eine Position nach links bewegt, andernfalls wird in das durch den Zeiger des vierten Bandes markierten Feld eine 0 geschrieben und beide Zeiger einen Schritt nach links bewegt. Nachdem dies für alle nichtleeren Felder des vierten Bandes geschehen ist, werden beide Zeiger wieder zurück in ihre Ausgangsposition gebracht. Indem wiederum alle nichtleeren Felder des vierten Bandes durchlaufen und darauf getestet werden, ob sie eine 1 oder eine 0 enthalten, wird die Ausgabe auf das Ausgabeband geschrieben und der Zeiger des Ausgabebandes an den Anfang der Ausgabe gesetzt. Da sich wieder durch Einfügen von „Springe in die nächste Zeile“-Anweisungen alle Teile auf die gleiche Länge bringen lassen, gibt es eine Konstante B, so daß die Simulation des Ausgabeknotens bei Eingabe von $\bar{x}$ und k vorher durchlaufenen Knoten genau $B \cdot (|\bar{x}| + k)$ Schritte benötigt.

Insgesamt gilt also für die so konstruierte Maschine $\mathbb{A}_{\mathbb{F}}$, für $\bar{x}, \bar{y} \in M^*$ und $T_{min} := \min\{t \mid \bar{x} \vdash_{\mathbb{F}}^{t} \bar{y}\}$:

$$\bar{x} \vdash_{\mathbb{F}}^{T_m in} \bar{y} \Leftrightarrow \mathbb{A}_{\mathbb{F}}(\bar{x}) = \bar{y} \text{ nach genau } (4 \cdot A + B) \cdot |\bar{x}| + (C_{\mathbb{F}} + B) \cdot T_{min} \text{ Schritten.}$$

Für die Polynome $p(n) := (4 \cdot A + B) \cdot n$ und $q_{\mathbb{F}}(n) := (C_{\mathbb{F}} + B) \cdot n$ gilt damit für alle $\bar{x}, \bar{y} \in M^*$ und alle $T \in \mathbb{N}$:

$$\bar{x} \vdash_{\mathbb{F}}^{T} \bar{y} \quad \Leftrightarrow \quad \begin{array}{l} \mathbb{A}_{\mathbb{F}} \text{ hält bei Eingabe von } \bar{x} \text{ nach höchstens} \\ p(|\bar{x}|) + q_{\mathbb{F}}(T) \text{ Schritten und gibt } \bar{y} \text{ aus.} \end{array}$$

□

Daß andererseits jedes Programm einer Turingmaschine als ein Flußdiagramm darstellbar ist, ist intuitiv klar. Die technische Schwierigkeit besteht hierbei darin, die mindestens zwei Bänder der Turingmaschine durch Elemente von M_∞ darzustellen.

Satz 3.1.8 („Für jede TM gibt es eine polynomiell äquivalente TMFD“) *Sei $\mathbb{A}$ eine deterministische Turingmaschine im Sinne einer Struktur $\mathcal{M}$. Dann gibt es eine deterministische Turingmaschine in Flußdiagrammdarstellung $\mathbb{F}_{\mathbb{A}}$ im Sinne von $\mathcal{M}$ und zwei Polynome $p_{\mathbb{A}}(n)$ und $q_{\mathbb{A}}(n)$, so daß für alle $\bar{x}, \bar{y} \in M^*$ und alle $T \in \mathbb{N}_{>0}$ gilt:*

- *Wenn $\mathbb{A}(\bar{x}) = \bar{y}$ nach höchstens T Schritten, so gilt $\bar{x} \vdash_{\mathbb{F}_{\mathbb{A}}}^{p_{\mathbb{A}}(|\bar{x}|)+q_{\mathbb{A}}(T)} \bar{y}$.*
- *Wenn $\bar{x} \vdash_{\mathbb{F}}^{T} \bar{y}$, so $\mathbb{A}(\bar{x}) = \bar{y}$ nach höchstens T Schritten.*

Beweis: Sei also $\mathbb{A}$ eine deterministische Turingmaschine im Sinne von $\mathcal{M}$. Um die Maschine $\mathbb{F}_{\mathbb{A}}$ zu konstruieren ersetzen wir intuitiv jede Programmzeile des Programms von $\mathbb{A}$ durch einen entsprechend gelabelten Knoten. Die Abfolge der Programmzeilen beziehungsweise die Sprungbefehle stellen wir über entsprechende Kanten dar.

Die technische Schwierigkeit hierbei ist nun die, daß wir die k Bänder von $\mathbb{A}$ mit ihren Feldinhalten durch Elemente von M_∞ darstellen müssen. Dies gelingt aber, indem wir in einem ersten Schritt die k Bänder „verschachteln“: Ein Element $(\ldots, x_{-1}, x_0.x_1, x_2, \ldots) \in M_\infty$ stellt dann für $i \in [k]$ die Feldinhalte des i-ten Bandes durch die $(i + k \cdot j)$-ten Komponenten für $j \in \mathbb{Z}$ dar. Der Inhalt des ersten Feldes des ersten Bandes zum Beispiel wird dann durch die Komponente x_1 dargestellt, der Inhalt des zweiten Feldes des ersten Bandes durch die Komponente x_{1+k}.

Um die Position der Zeiger von $\mathbb{A}$ darzustellen, fügen wir im zweiten Schritt für jedes Feld eine Markierung ein, welche sagt, ob über dem entsprechenden Feld ein Zeiger steht. Somit ist die $(2i+2k\cdot j)$-te Komponente von $(\ldots, x_{-1}, x_0.x_1, x_2, \ldots) \in M_\infty$ nun der Inhalt eines Feldes des i-ten Bandes, und die $(2i+2k\cdot j-1)$-te Komponente ist 1, falls der Zeiger des i-ten Bandes über diesem Feld steht, andernfalls 0.

Im dritten Schritt sorgen wir dafür, daß für jedes Feld kodiert wird, ob es leer ist. Hierfür fügen wir wieder wie im zweiten Schritt eine Markierung ein. Nun ist für ein $(\ldots, x_{-1}, x_0.x_1, x_2, \ldots) \in M_\infty$ die $(3i + 3k \cdot j)$-te Komponente der Inhalt eines Feldes des i-ten Bandes, die $(3i + 3k \cdot j - 1)$-te Komponente kodiert, ob der Zeiger des i-ten Bandes über diesem Feld steht, und die $(3i + 3k \cdot j - 2)$-te Komponente ist 0, falls das Feld leer ist, andernfalls 1.

Nachdem nun die Feldinhalte durch ein Element von M_∞ dargestellt werden können, fügen wir einen Zwischenspeicher ein, um aus den Feldinhalten eines Bandes berechnete Funktionswerte auf ein anderes Band schreiben zu können, und einen weiteren Zwischenspeicher, den wir aus technischen Gründen benötigen. Hierzu benutzen wir die 0-te und die (-1)-te Komponente. Für $j < 0$ beschreiben nun die $(3i+3k\cdot j-2)$-ten, $(3i+3k\cdot j-3)$-ten und $(3i + 3k \cdot j - 4)$-ten Komponenten die entsprechenden Felder des i-ten Bandes. Für eine graphische Darstellung siehe Abbildung 3.2.

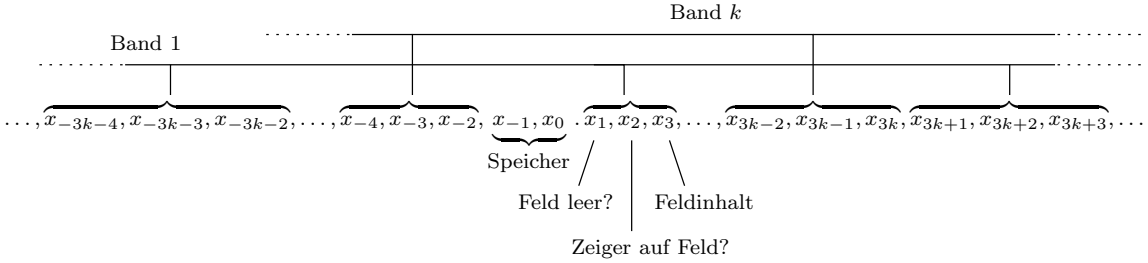


Abbildung 3.2: Die Darstellung der Bänder einer Turingmaschine durch ein Element aus M_∞

Um die Eingabe in die oben dargestellte Form zu bringen, verwenden wir die Subroutine $\texttt{Init}_k$. Diese konstruiert bei Eingabe von $\bar{x} \in M^*$ die Darstellung der k Bänder, wobei $\bar{x}$ in den Feldern des Eingabebandes steht, und der Zeiger des Eingabebandes das erste der $\bar{x}$ enthaltenden Felder markiert. Siehe hierzu Abbildung 3.3. Zunächst wird da-

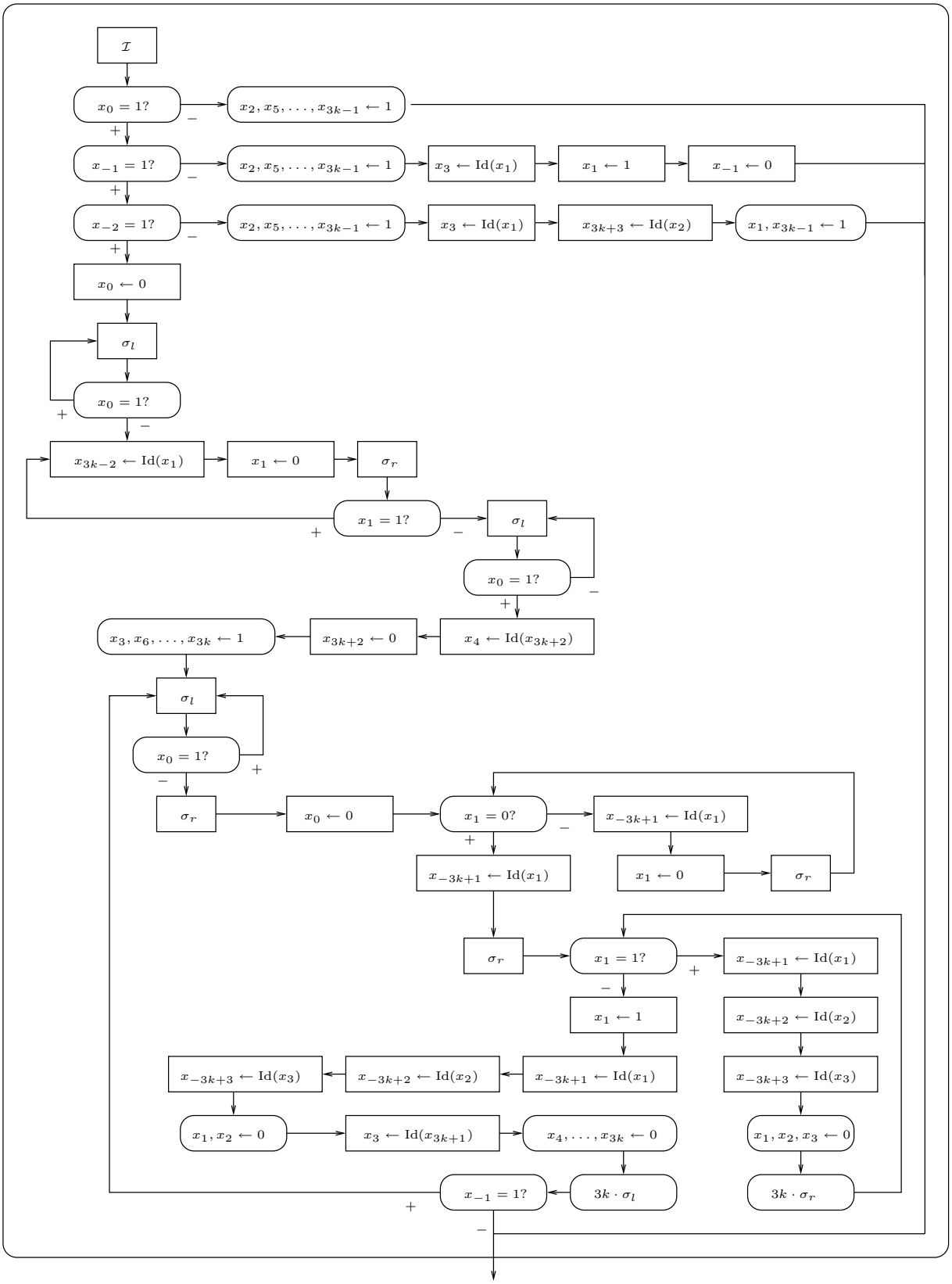


Abbildung 3.3: Die Subroutine $\texttt{Init}_k$

bei überprüft, ob die Länge der Eingabe kleiner als drei ist. Die entsprechenden Fälle werden separat behandelt. Ist $|\bar{x}| \geq 3$, so wird zunächst eine Markierung eingefügt, mit deren Hilfe der Beginn der Eingabe wiedergefunden werden kann. Daraufhin werden die Elemente der Eingabe sukzessive auseinandergezogen, wobei jeweils die Markierungen, welche anzeigen, daß das entsprechende Feld nicht leer ist, eingefügt werden. Außerdem werden noch die Markierungen für die Positionen der Zeiger eingefügt.

Für den Fall, daß $|\bar{x}| \leq 2$, und das Auseinanderziehen des ersten Elementes der Eingabe werden für entsprechende Konstanten B_k und C_k höchstens $B_k \cdot |\bar{x}| + C_k$ viele Zustandswechsel benötigt. Um den Rest der Eingabe in die richtige Form zu bringen sind mit der entsprechenden Konstante A_k für jedes einzelne der $|\bar{x}| - 1$ übrigen Elemente $A_k \cdot |\bar{x}|$ viele Zustandswechsel nötig. Insgesamt werden bei Eingabe von $\bar{x}$ also mit $p_{init_k}(n) := A_k \cdot n^2 + B_k \cdot n + C_k$ höchstens $p_{init_k}(|\bar{x}|)$ viele Zustandswechsel für die Subroutine $\texttt{Init}_k$ benötigt.

Da wir den Zwischenspeicher immer auf der 0-ten und (-1)-ten Komponente behalten wollen, können wir nicht mehr einfach die Shiftabbildungen σ_l und σ_r verwenden. Stattdessen verwenden wir die in Abbildung 3.4 dargestellten Subroutinen σ'_l und σ'_r, welche unter Benutzung des Zwischenspeichers in der (-1)-ten Komponente den Inhalt des zweiten Zwischenspeichers mitbewegen.

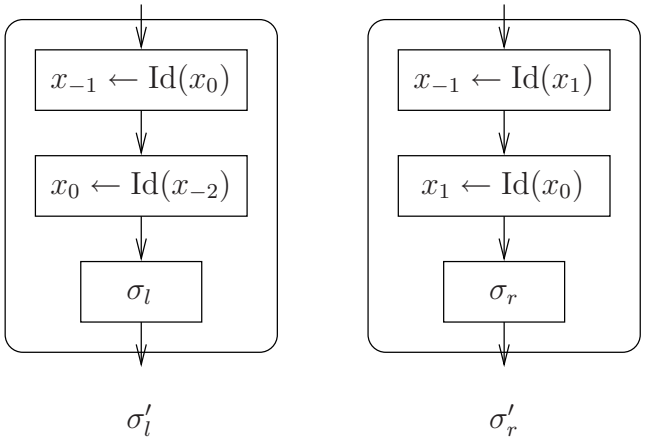
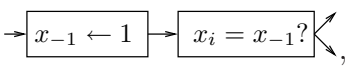


Abbildung 3.4: Die Subroutinen σ'_l und σ'_r

Desweiteren benötigen wir noch die Subroutine $\texttt{ptrpos}_k\texttt{(}i\texttt{)}$, welche dafür sorgt, daß die $3i$-te Komponente gerade der Inhalt des durch den Zeiger des i-ten Bandes markierten Feldes ist. Hierzu genügt es, soweit nach links zu gehen, bis das Ende der beschriebenen Felder des i-ten Bandes erreicht ist, und danach solange wieder nach rechts zu gehen, bis die Markierung des Zeigers des i-ten Bandes erreicht ist. Die Subroutine $\texttt{ptrpos}_k\texttt{(}i\texttt{)}$ ist dargestellt in Abbildung 3.5.

Um die Darstellung lesbarer zu machen schreiben wir für zwei Knoten



,

welche mit Hilfe des Zwischenspeichers in der (-1)-ten Komponente überprüfen, ob die

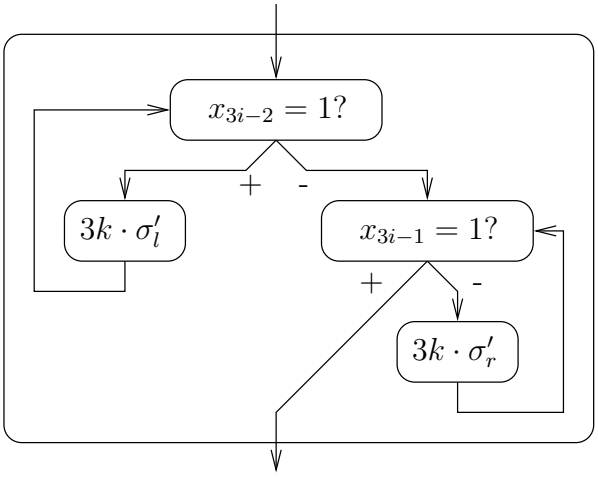
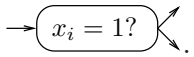


Abbildung 3.5: Die Subroutine $\texttt{ptrpos}_k(i)$

i-te Komponente 1 ist, die Subroutine



Im Fall, daß überprüft wird, ob die i-te Komponente 0 ist, analog.

Nun haben wir die technischen Hilfsmittel beisammen, die für die Konstruktion von $\mathbb{F}_{\mathbb{A}}$ nötig sind. Hierfür beginnen wir mit der Subroutine $\texttt{Init}_k$, um die Eingabe in die richtige Form zu bringen. Nun fügen wir sukzessive für jede Zeile des Programms von $\mathbb{A}$ eine entsprechende Subroutine hinzu. Für eine Zeilennummer q sei η_q der erste Knoten der entsprechenden Subroutine. Die von den Subroutine $\texttt{Init}_k$ ausgehende Kante endet an dem Knoten η_1.

- Falls Π_q=`(move` i `left)` oder Π_q=`(move` i `right)`, so fügen wir die Subroutine $\texttt{ptrpos}_k(i)$, einen mit $x_{-(3k+2)+(3i-1)} \leftarrow 1$ beziehungsweise $x_{6i-1} \leftarrow 1$ gelabelten Knoten und einen mit $x_{3i-1} \leftarrow 0$ gelabelten Knoten hinzu (siehe Abbildung 3.6). Die ausgehende Kante endet an dem Knoten η_{q+1}.

- Falls Π_q =`(compute` f, i `to` j`)` für eine n-stellige Funktion f von $\mathcal{M}$ und $i, j \in [k]$, so fügen wir die Subroutine $\texttt{ptrpos}_k(i)$, einen mit $x_0 \leftarrow f(x_{3i}, x_{3i+3k}, \ldots, x_{3i+(n-1)\cdot 3k})$ gelabelten Knoten, die Subroutine $\texttt{ptrpos}_k(j)$ und einen mit $x_{3j} \leftarrow Id(x_0)$ gelabelten Knoten hinzu (siehe Abbildung 3.7). Die ausgehende Kante endet an dem Knoten η_{q+1}.

- Falls Π_q =`(if` R, i `then` p`, else` p'`)` für eine n-stellige Relation von $\mathcal{M}$ und $p, p' \in [k]$, so fügen wir die Subroutine $\texttt{ptrpos}_k(i)$ und einen mit $Rx_{3i}x_{3i+3k}\ldots x_{3i+(n-1)\cdot 3k}?$ gelabelten Knoten $\eta_{q?}$ ein. Die von $\eta_{q?}$ ausgehenden Kanten enden an den Knoten $\beta^+(\eta_{q?}) = \eta_p$ und $\beta^-(\eta_{q?}) = \eta_{p'}$ (siehe Abbildung 3.8).

- Falls Π_q =`(if` $i = \lambda$ `then` p`, else` p'`)` für $i \in [k]$, so fügen wir wieder die Subroutine $\texttt{ptrpos}_k(i)$, einen mit $x_{-1} \leftarrow 0$ gelabelten Knoten und einen mit $x_{-1} = x_{3i-2}?$

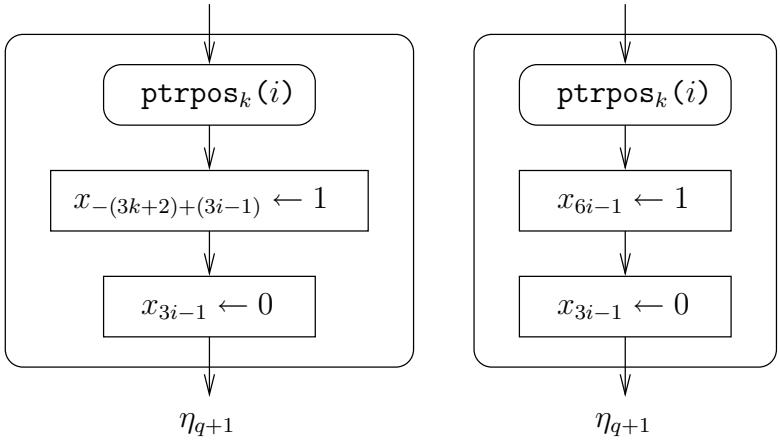


Abbildung 3.6: Die Subroutinen für Zeilen der Form $(q,$`(move` i `left)`$)$ beziehungsweise $(q,$`(move` i `right)`$)$

gelabelten Knoten $\eta_{q?}$ ein. Die von $\eta_{q?}$ ausgehenden Kanten enden an den Knoten $\beta^+(\eta_{q?}) = \eta_p$ und $\beta^-(\eta_{q?}) = \eta_{p'}$ (siehe Abbildung 3.8).

- Falls $\Pi_q =$`(stop)`, so fügen wir die in Abbildung 3.9 dargestellte Subroutine `output`$_k$ an, welche bewirkt, daß das richtige Ergebnis ausgegeben wird. Hierfür wird zunächst überprüft, ob die Ausgabe die Länge 0 oder 1 hat. Diese Fälle werden gesondert behandelt. Andernfalls werden alle für die Berechnung nicht mehr wichtigen Komponenten auf 0 gesetzt. Anschließend werden eine nach der anderen die für die Ausgabe relevanten Komponenten zusammengeschrieben und die entsprechende Anzahl von Einsen in die jeweiligen Komponenten geschrieben. Als letztes wird die Ausgabe mit Hilfe der Outputabbildung ausgegeben. Da die für die Subroutine nur diejenigen Komponenten wichtig sind, welche die Felder des k-ten Bandes der Turingmaschine kodieren, hängt die Anzahl der benötigten Zustandswechsel außer von k nur von der Anzahl der beschriebenen Felder des k-ten Bandes ab. Sei t die Anzahl der beschriebenen Felder des k-ten Bandes. Um die nicht relevanten Komponenten auf 0 zu setzen, werden für eine Konstante C_1 gerade $k \cdot t \cdot C_1$ Zustandswechsel benötigt. Für das Zusammenschreiben der t relevanten Komponenten sind dann für eine Konstante C_2 noch höchstens $t \cdot (k \cdot t \cdot C_2)$ Zustandswechsel notwendig. Die gesonderten Fälle benötigen für eine entsprechende Konstante gerade C_3 Zustandswechsel. Insgesamt sind also mit $p_{output_k}(n) := kC_2 \cdot n^2 + kC_1 \cdot n + C_3$ bei t beschriebenen Feldern des k-ten Bandes höchstens $p_{output_k}(t)$ Zustandswechsel notwendig.

Die so konstruierte deterministische Turingmaschine in Flußdiagrammdarstellung $\mathbb{F}_\mathbb{A}$ leistet nun nach Konstruktion dasselbe, wie die Maschine $\mathbb{A}$, es gilt für alle $\bar{x}, \bar{y} \in M^*$:

$$\text{Es gibt ein } t \in \mathbb{N} \text{ mit } \bar{x} \vdash^t_{\mathbb{F}_\mathbb{A}} \bar{y} \Leftrightarrow \mathbb{A}(\bar{x}) = \bar{y}.$$

Betrachten wir die benötigte Zeit. Für $\bar{x}, \bar{y} \in M^*$ und $T \in \mathbb{N}_{>0}$ mit $\mathbb{A}(\bar{x}) = \bar{y}$ nach T Schritten benötigt die Subroutine `init`$_k$ höchstens $p_{init_k}(|\bar{x}|)$ viele Zustandswechsel. Da die Maschine $\mathbb{A}$ nur T Schritte lang läuft, sind zu jedem Zeitpunkt der Berechnung auf jedem

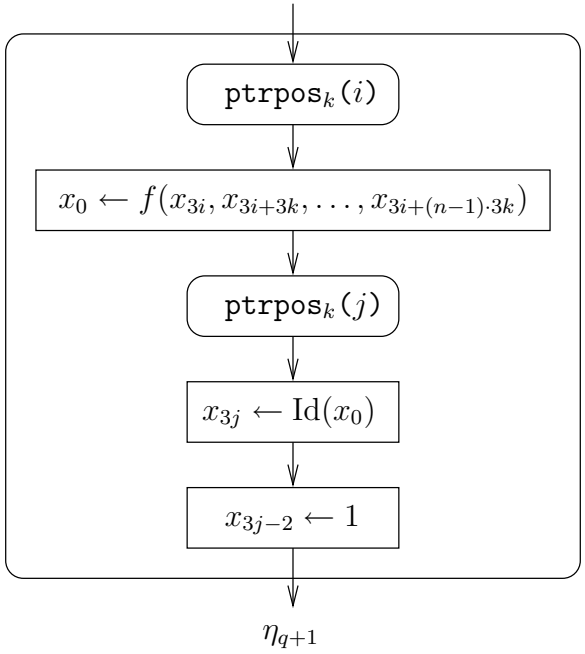


Abbildung 3.7: Die Subroutine für eine Zeile der Form $(q,$`(compute` f, i `to` $j)$`)`

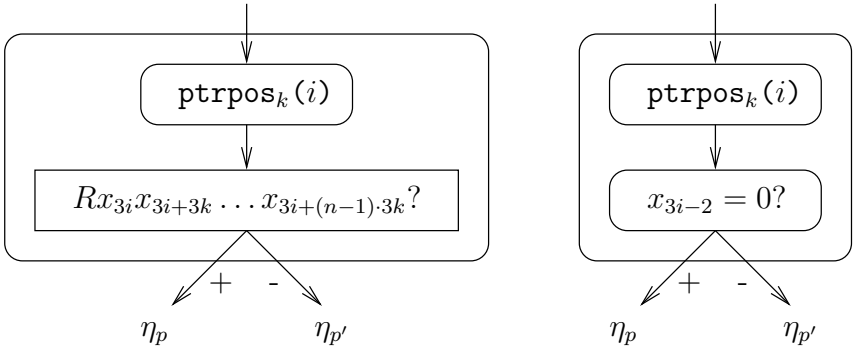


Abbildung 3.8: Die Subroutinen für die Verzweigungszeilen

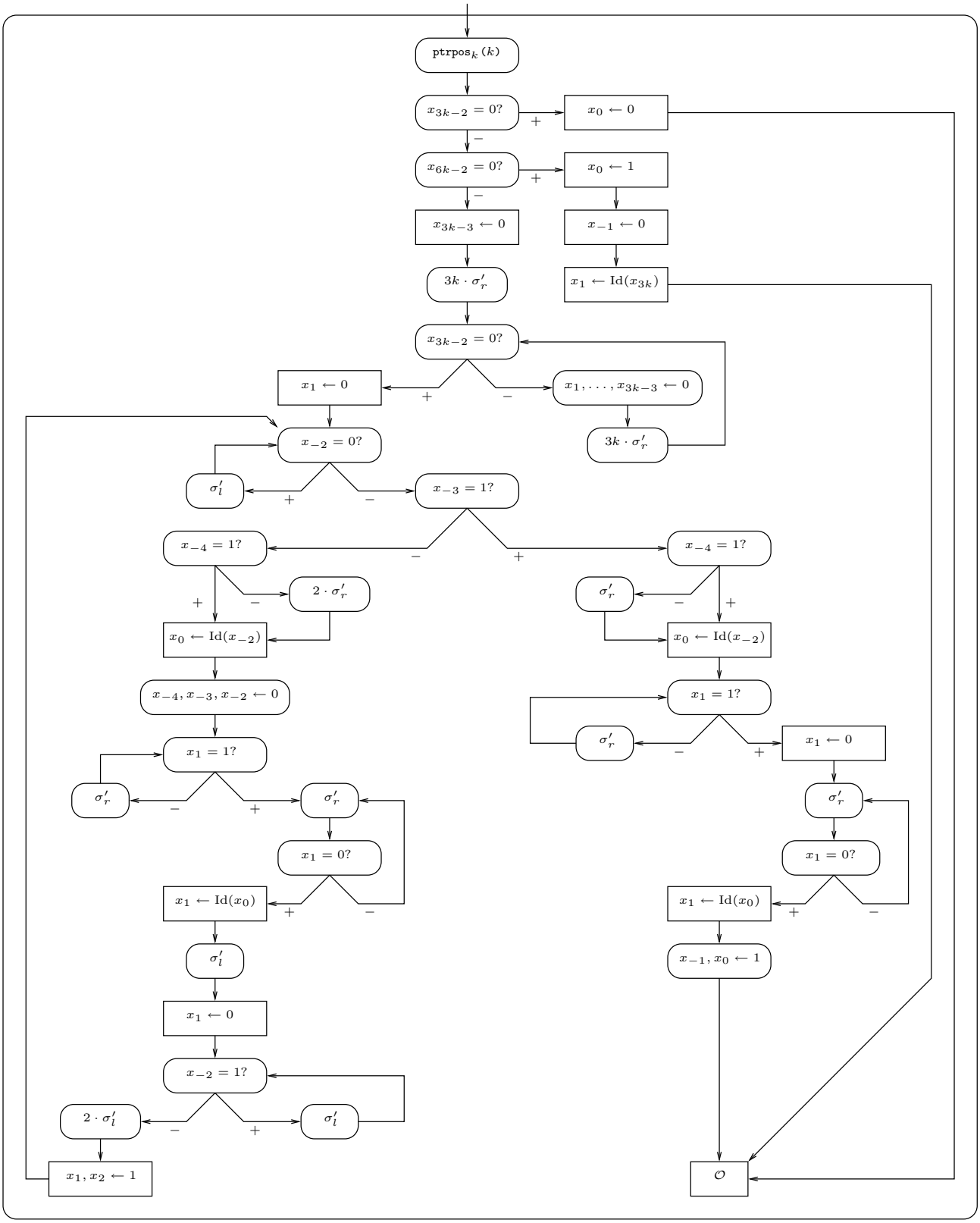


Abbildung 3.9: Die Subroutine $\texttt{output}_k$ für den Befehl `(stop)`

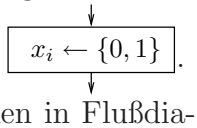
ihrer Bänder auch nur höchstens T Felder nicht leer. Für jeden Aufruf einer Subroutine $\texttt{ptrpos}_k(i)$ benötigt $\mathbb{F}_\mathbb{A}$ also für eine geeignete Konstante C_{ptrpos_k} höchstens $2T \cdot C_{ptrpos_k}$ Zustandswechsel. Für jede bis auf die letzte von $\mathbb{A}$ durchlaufene Programmzeile besteht die entsprechende Subroutine von $\mathbb{F}_\mathbb{A}$ aus höchstens zwei Aufrufen von $\texttt{ptrpos}_k$ und höchstens drei weiteren Knoten. Somit sind für jede von $\mathbb{A}$ durchlaufene Programmzeile (bis auf die letze Zeile) höchstens $4T \cdot C_{ptrpos_k} + 3$ Zustandswechsel von $\mathbb{F}_\mathbb{A}$ erforderlich. Für die Subroutine $\texttt{output}_k$ schließlich sind, da höchstens T Felder des k-ten Bandes von $\mathbb{A}$ nicht leer sind, höchstens $p_{output_k}(T)$ Zustandswechsel notwendig.

Insgesamt sind somit $p_{init_k}(|\bar{x}|)+T\cdot(4T\cdot C_{ptrpos_k}+3)+p_{output_k}(T)$ viele Zustandswechsel erforderlich. Mit $p_\mathbb{A}(n) := p_{init_k}(n)$ und $q_\mathbb{A}(n) := n \cdot (4n \cdot C_{ptrpos_k} + 3) + p_{output_k}(n)$ gilt also $\bar{x} \vdash_{\mathbb{F}_\mathbb{A}}^{p_\mathbb{A}(|\bar{x}|)+q_\mathbb{A}(T)} \bar{y}$.

Andererseits simuliert $\mathbb{F}_\mathbb{A}$ mit t Zustandswechseln höchstens t Schritte von $\mathbb{A}$. Somit gilt für $\bar{x}, \bar{y} \in M^*$ und $T \in \mathbb{N}$ mit $\bar{x} \vdash_{\mathbb{F}_\mathbb{A}}^{T} \bar{y}$, daß $\mathbb{A}$ bei Eingabe von $\bar{x}$ nach höchstens T Schritten hält und $\bar{y}$ ausgibt. □

3.2 Nichtdeterministische Maschinen

Analog zu der bei der Behandlung der nichtdeterministischen Turingmaschinen über einer Struktur $\mathcal{M}$ eingeführten Unterscheidung zwischen binärem und vollständigem Nichtdeterminismus betrachten wir auch bei den nichtdeterministischen Turingmaschinen in Flußdiagrammdarstellung die binär nichtdeterministische und die vollständig nichtdeterministische Version. Zunächst zu der binär nichtdeterministischen Version.

Definition 3.2.1 (BNTMFD/$\mathcal{M}$) Sei $\mathcal{M}$ eine Struktur. Eine *binär nichtdeterministische Turingmaschine in Flußdiagrammdarstellung über* $\mathcal{M}$ ist eine deterministische Turingmaschine in Flußdiagrammdarstellung über $\mathcal{M}$, welche zusätzlich zu den Eingangsknoten, Shiftknoten, Berechnungsknoten, Verzweigungsknoten und dem Ausgangsknoten noch *binäre Rateknoten* enthalten kann. Diese haben mindestens eine eingehende und genau eine ausgehende Kante und sind mit $x_i \leftarrow \{0,1\}$ für $i \in \mathbb{N}$ gelabelt: 

.

Analog zu den bei der Definition der deterministischen Turingmaschinen in Flußdiagrammdarstellung über $\mathcal{M}$ eingeführten Bezeichnungen ist auch bei einer binär nichtdeterministischen Turingmaschine $\mathbb{F}$ in Flußdiagrammdarstellung über $\mathcal{M}$ die Menge der Knoten von $\mathbb{F}$ bezeichnet mit $\mathcal{K}_\mathbb{F}$. Der eindeutige Eingangsknoten wird mit η_{in}, der eindeutige Ausgangsknoten mit η_{out} bezeichnet. Falls η ein Verzweigungsknoten ist, so sind die beiden Knoten, an denen die beiden von η ausgehenden Kanten enden, wieder zusätzlich mit $+$ und $-$ gelabelt und werden mit $\beta^+(\eta)$ beziehungsweise $\beta^-(\eta)$ bezeichnet. Ist $\eta = \eta_{out}$, so ist $\beta(\eta) = \eta$. Falls η kein Verzweigungs- und kein Ausgangsknoten ist, so bezeichne $\beta(\eta)$ den eindeutigen Knoten, an dem die von η ausgehende Kante endet.

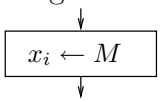
Wie im deterministischen Fall definieren wir wieder die Nachfolgerelation $\vdash_\mathbb{F}$:

Definition 3.2.2 ($\vdash_\mathbb{F}$ für BNTMFD/$\mathcal{M}$) Sei $\mathbb{F}$ eine binär nichtdeterministische Turingmaschine in Flußdiagrammdarstellung über einer Strukur $\mathcal{M}$. Die *Nachfolgerelation*

$\vdash_{\mathbb{F}} \subseteq (\mathcal{K}_{\mathbb{F}} \times M_\infty) \times (\mathcal{K}_{\mathbb{F}} \times M_\infty)$ wird analog zur Nachfolgerelation für deterministische Turingmaschinen in Flußdiagrammdarstellung definiert. Hierbei kann ein zusätzlicher Fall eintreten:

- Für $(\eta_1, v), (\eta_2, w) \in \mathcal{K}_{\mathbb{F}} \times M_\infty$ gilt $(\eta_1, v) \vdash_{\mathbb{F}} (\eta_2, w)$, falls η_1 mit $x_i \leftarrow \{0,1\}$ gelabelt ist und $\eta_2 = \beta(\eta_1)$ und $w_i \in \{0,1\}$ und $w_k = v_k$ für $k \neq i$.

Wie bei den vollständig nichtdeterministischen Turingmaschinen erlauben wir auch bei vollständig nichtdeterministischen Turingmaschinen in Flußdiagrammdarstellung, daß in den Rateknoten nicht nur ein Element aus $\{0,1\}$, sondern aus M geraten wird:

Definition 3.2.3 (NTMFD/$\mathcal{M}$) Eine *vollständig nichtdeterministische Turingmaschine in Flußdiagrammdarstellung über einer Struktur* $\mathcal{M}$ ist eine deterministische Turingmaschine in Flußdiagrammdarstellung über $\mathcal{M}$, die zusätzlich noch *Rateknoten* enthalten kann. Ein Rateknoten hat mindestens eine eingehende und genau eine ausgehende Kante und ist mit $x_i \leftarrow M$ für ein $i \in \mathbb{N}$ gelabelt: 

. Ist η ein Rateknoten, so wird der eindeutige nächste Knoten mit $\beta(\eta)$ bezeichnet. Auf den anderen Knoten sind β, β^+, β^- wie gewohnt definiert.

Auch hier definieren wir wieder die Nachfolgerelation:

Definition 3.2.4 ($\vdash_{\mathbb{F}}$ für NTMFD/$\mathcal{M}$) Die *Nachfolgerelation* $\vdash_{\mathbb{F}} \subseteq (\mathcal{K}_{\mathbb{F}} \times M_\infty) \times (\mathcal{K}_{\mathbb{F}} \times M_\infty)$ einer vollständig nichtdeterministischen Turingmaschine in Flußdiagrammdarstellung $\mathbb{F}$ über einer Strukur $\mathcal{M}$ ist analog zur Nachfolgerelation für deterministische Turingmaschinen in Flußdiagrammdarstellung definiert. Es kann wieder ein zusätzlicher Fall eintreten:

- Für $(\eta_1, v), (\eta_2, w) \in \mathcal{K}_{\mathbb{F}} \times M_\infty$ gilt $(\eta_1, v) \vdash_{\mathbb{F}} (\eta_2, w)$, falls η_1 mit $x_i \leftarrow M$ gelabelt ist und $\eta_2 = \beta(\eta_1)$ und $w_i \in M$ und $w_k = v_k$ für $k \neq i$.

Analog zum deterministischen Fall betrachten wir auch bei binär nichtdeterministischen und vollständig nichtdeterministischen Turingmaschinen in Flußdiagrammdarstellung wieder die Zeit-T-Eingabe-Ausgabe-Relation.

Definition 3.2.5 ($\vdash_{\mathbb{F}}^T$ für BNTMFD / NTMFD) Sei $\mathbb{F}$ eine binär oder vollständig nichtdeterministische Turingmaschine in Flußdiagrammdarstellung über einer Struktur $\mathcal{M}$. Für $T \in \mathbb{N}_{>=}$ wird die *Zeit-T-Eingabe-Ausgabe-Relation* $\vdash_{\mathbb{F}}^T \subseteq M^* \times M^*$ von $\mathbb{F}$ definiert durch:

Für $\bar{x}, \bar{y} \in M^*$ und $T \in \mathbb{N}_{>0}$ gilt $\bar{x} \vdash_{\mathbb{F}}^T \bar{y}$ genau dann, wenn $z^0, \ldots, z^T \in \mathcal{K}_{\mathbb{F}} \times M_\infty$ existieren, so daß

- $z^0 = (\eta_{in}, \mathcal{I}(\bar{x}))$
- $z^T = (\eta_{out}, z)$ mit $\mathcal{O}(z) = \bar{y}$
- für alle $i \in [T]$ gilt $z^{i-1} \vdash_{\mathbb{F}} z^i$.

3.2.1 Der Zusammenhang zwischen nichtdeterministischen Turingmaschinen und nichtdeterministischen Turingmaschinen in Flußdiagrammdarstellung

Wie im deterministischen Fall läßt sich ein Zusammenhang zwischen binär nichtdeterminstischen Turingmaschinen in Flußdiagrammdarstellung und binär nichtdeterminstischen Turingmaschinen zeigen. Den Hauptteil der Arbeit hierfür haben wir schon im determinstischen Fall erledigt.

Satz 3.2.6 (Äquivalenz von BNTM und BNTMFD) *Sei $\mathcal{M}$ eine Struktur.*

(i) Sei $\mathbb{F}$ eine binär nichtdeterministische Turingmaschine in Flußdiagrammdarstellung über $\mathcal{M}$. Dann gibt es eine binär nichtdeterministische Turingmaschine $\mathbb{A}_{\mathbb{F}}$ über $\mathcal{M}$ und Polynome $p(n)$ und $q_{\mathbb{F}}(n)$, so daß für alle $\bar{x}, \bar{y} \in M^*$ und jedes $T \in \mathbb{N}_{>0}$ gilt:

$$\bar{x} \vdash_{\mathbb{F}}^{T} \bar{y} \quad \Leftrightarrow \quad \begin{array}{l} \text{es gibt bei Eingabe von } \bar{x} \text{ eine Berechnung von } \mathbb{A}_{\mathbb{F}} \text{ mit} \\ \text{höchstens } p(|\bar{x}|) + q_{\mathbb{F}}(T) \text{ Schritten, welche } \bar{y} \text{ ausgibt.} \end{array}$$

(ii) Sei $\mathbb{A}$ eine binär nichtdeterminstische Turingmaschine über $\mathcal{M}$. Dann gibt es eine binär nichtdeterministische Turingmaschine in Flußdiagrammdarstellung $\mathbb{F}_{\mathbb{A}}$ über $\mathcal{M}$ und Polynome $p_{\mathbb{A}}(n)$ und $q_{\mathbb{A}}(n)$, so daß für alle $\bar{x}, \bar{y} \in M^*$ und jedes $T \in \mathbb{N}_{>0}$ gilt:

- *Wenn es bei Eingabe von $\bar{x}$ eine Berechnung von $\mathbb{A}$ mit höchstens T Schritten und Ausgabe $\bar{y}$ gibt, so gilt $\bar{x} \vdash_{\mathbb{F}_{\mathbb{A}}}^{p_{\mathbb{A}}(|\bar{x}|)+q_{\mathbb{A}}(T)} \bar{y}$.*
- *Wenn $\bar{x} \vdash_{\mathbb{F}_{\mathbb{A}}}^{T} \bar{y}$ gilt, so gibt es bei Eingabe von $\bar{x}$ eine Berechnung von $\mathbb{A}$ mit höchstens T Schritten, welche $\bar{y}$ ausgibt.*

Beweis: *Zu (i):* Sei $\mathbb{F}$ eine binär nichtdeterministische Turingmaschine in Flußdiagrammdarstellung über $\mathcal{M}$. Die binär nichtdeterministische Turingmaschine $\mathbb{A}_{\mathbb{F}}$ über $\mathcal{M}$ mit fünf Bändern konstruieren wir analog zu der im Beweis von Satz 3.1.7 konstruierten deterministischen Turingmaschine:

Die Maschine $\mathbb{A}_{\mathbb{F}}$ berechnet bei Eingabe von $\bar{x}$ zunächst die Länge von $\bar{x}$, speichert diese auf dem vierten Band und stellt $\mathcal{I}(\bar{x})$ auf dem zweiten Band dar. Dann werden nach den in $\mathbb{F}$ dargestellten Anweisungen nacheinander die den jeweiligen Knoten entsprechenden Blöcke von Programmzeilen aufgerufen. Hierbei kann nun ein weiterer Fall eintreten:

- Falls der Knoten η mit $x_i \leftarrow \{0, 1\}$ gelabelt ist, so wird der Zeiger des zweiten Bandes i Schritte nach rechts bewegt, mit Hilfe einer binären Rateanweisung 0 oder 1 in das unter diesem Zeiger liegende Feld geschrieben, und der Zeiger wieder i Schritte nach links bewegt. Danach wird der Zähler für die durchlaufenen Knoten um eins erhöht und das Programm springt in die Zeile, mit der der zum Knoten $\beta(\eta)$ gehörige Block anfängt.

Auch die Anzahl der für einen solchen Block benötigten Schritte kann durch Einfügen von „Springe in die nächste Zeile“-Anweisungen auf die Anzahl der für die übrigen Blöcke

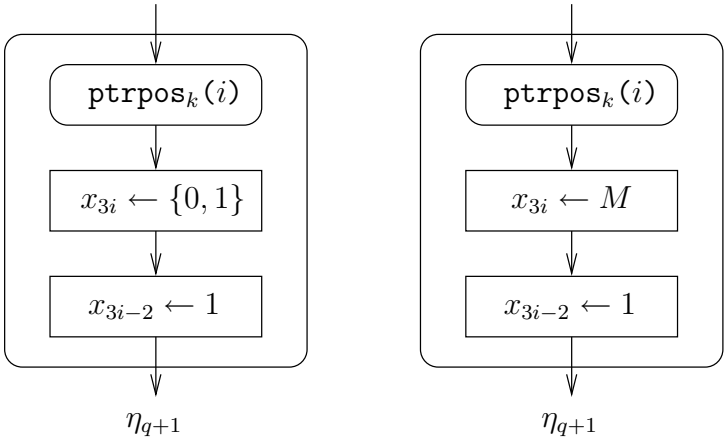


Abbildung 3.10: Die Subroutinen für die binären Ratebefehle und die Ratebefehle

benötigten Schritte gebracht werden. Die Outputabbildung wird genauso wie im deterministischen Fall berechnet.

Die Berechnung läuft also für die Eingabe $\bar{x}$ und k durchlaufene Knoten wieder $4 \cdot A \cdot |\bar{x}|$ Schritte für die Inputabbildung, $C_{\mathbb{F}} \cdot k$ Schritte für die k Knoten und $B \cdot (|\bar{x}| + k)$ Schritte für die Outputabbildung. Mit $p(n) := (4 \cdot A + B) \cdot n$ und $q_{\mathbb{F}}(n) := (C_{\mathbb{F}} + B) \cdot n$ folgt die Behauptung.

Zu (ii): Sei $\mathbb{A}$ eine binär nichtdeterministische Turingmaschine über $\mathcal{M}$ mit k Bändern. Wir konstruieren die binär nichtdeterminstische Turingmaschine in Flußdiagrammdarstellung $\mathbb{F}_{\mathbb{A}}$ über $\mathcal{M}$ analog zu der im Beweis von Satz 3.1.8 konstruierten deterministischen Turingmaschine in Flußdiagrammdarstellung:

Zunächst fügen wir die Subroutine $\texttt{Init}_k$ ein, um die k Bänder von $\mathbb{A}$ in M_∞ darzustellen (siehe Abbildung 3.3). Dann fügen wir wie im deterministischen Fall für jede Programmzeile q von $\mathbb{A}$ eine entsprechende Subroutine beginnend mit dem Knoten η_q ein. Hierbei kann nun ein zusätzlicher Fall eintreten:

- Falls $\Pi_q =$ `(bin-guess` i`)` für $i \in [k]$, so fügen wir die Subroutine $\texttt{ptrpos}_k(i)$, einen mit $x_{3i} \leftarrow \{0,1\}$ gelabelten Knoten und einen mit $x_{3i-2} \leftarrow 1$ gelabelten Knoten hinzu. Die ausgehende Kante endet an dem Knoten η_{q+1} (siehe Abbildung 3.10).

Wiederum werden für die Subroutine $\texttt{Init}_k$ höchstens $p_{init_k}(|\bar{x}|)$ viele Zustandswechsel, für jede der höchstens T durchlaufenen Subroutinen für die Programmzeilen höchstens $4T \cdot C_{ptrpos_k} + 3$ viele und für die Subroutine $\texttt{output}_k$ höchstens $p_{output_k}(T)$ viele Zustandswechsel benötigt.

Wenn es eine Berechnung von $\mathbb{A}$ bei Eingabe von $\bar{x}$ mit höchstens T Schritten gibt, welche $\bar{y}$ ausgibt, so gibt es nach Konstruktion von $\mathbb{F}_{\mathbb{A}}$ also für $p_{\mathbb{A}}(n) := p_{init_k}(n)$ und $q_{\mathbb{A}}(n) := n \cdot (4n \cdot C_{ptrpos_k} + 3) + p_{output_k}(n)$ Zustände $z^0, \ldots, z^{p_{\mathbb{A}}(|\bar{x}|)+q_{\mathbb{A}}(T)} \in \mathcal{K}_{\mathbb{F}_{\mathbb{A}}} \times M_\infty$, so daß

- $z^0 = (\eta_1, \mathcal{I}(\bar{x}))$

- $z^{p_{\mathbb{A}}(|\bar{x}|)+q_{\mathbb{A}}(T)} = (\eta_{out}, z)$ mit $z = \bar{y}$
- $z^{i-1} \vdash_{\mathbb{F}_{\mathbb{A}}} z^i$ für alle $i \in [p_{\mathbb{A}}(|\bar{x}|) + q_{\mathbb{A}}(T)]$.

Nach Defintion gilt somit $\bar{x} \vdash_{\mathbb{F}_{\mathbb{A}}}^{p_{\mathbb{A}}(|\bar{x}|)+q_{\mathbb{A}}(T)} \bar{y}$.

Wenn für $\bar{x}, \bar{y} \in M^*$ und $T \in \mathbb{N}_{>0}$ andererseits $\Psi_{\mathbb{F}_{\mathbb{A}}}^T(\bar{x}, \bar{y})$ gilt, so gibt es nach Konstruktion von $\mathbb{F}_{\mathbb{A}}$ eine Berechnung von $\mathbb{A}$ bei Eingabe von $\bar{x}$ mit höchstens T Schritten, welche $\bar{y}$ ausgibt. □

Die analoge Aussage gilt auch für den vollständig nichtdeterministischen Fall, das Prinzip ist dasselbe wie im binär nichtdeterministischen Fall.

Satz 3.2.7 (Äquivalenz von NTM und NTMFD) *Sei $\mathcal{M}$ eine Struktur.*

(i) Sei $\mathbb{F}$ eine vollständig nichtdeterministische Turingmaschine in Flußdiagrammdarstellung über $\mathcal{M}$. Dann gibt es eine vollständig nichtdeterministische Turingmaschine $\mathbb{A}_{\mathbb{F}}$ über $\mathcal{M}$ und Polynome $p(n)$ und $q_{\mathbb{F}}(n)$, so daß für alle $\bar{x}, \bar{y} \in M^*$ und jedes $T \in \mathbb{N}_{>0}$ gilt:

$$\bar{x} \vdash_{\mathbb{F}}^T \bar{y} \quad \Leftrightarrow \quad \begin{array}{l} \text{es gibt bei Eingabe von } \bar{x} \text{ eine Berechnung von } \mathbb{A}_{\mathbb{F}} \text{ mit} \\ \text{höchstens } p(|\bar{x}|) + q_{\mathbb{F}}(T) \text{ Schritten, welche } \bar{y} \text{ ausgibt.} \end{array}$$

(ii) Sei $\mathbb{A}$ eine vollständig nichtdeterminstische Turingmaschine über $\mathcal{M}$. Dann gibt es eine vollständig nichtdeterministische Turingmaschine in Flußdiagrammdarstellung $\mathbb{F}_{\mathbb{A}}$ über $\mathcal{M}$ und Polynome $p_{\mathbb{A}}(n)$ und $q_{\mathbb{A}}(n)$, so daß für alle $\bar{x}, \bar{y} \in M^*$ und jedes $T \in \mathbb{N}_{>0}$ gilt:

- *Wenn es bei Eingabe von $\bar{x}$ eine Berechnung von $\mathbb{A}$ mit höchstens T Schritten und Ausgabe $\bar{y}$ gibt, so gilt $\bar{x} \vdash_{\mathbb{F}_{\mathbb{A}}}^{p_{\mathbb{A}}(|\bar{x}|)+q_{\mathbb{A}}(T)} \bar{y}$.*
- *Wenn $\bar{x} \vdash_{\mathbb{F}_{\mathbb{A}}}^T \bar{y}$, so gibt es bei Eingabe von $\bar{x}$ eine Berechnung von $\mathbb{A}$ mit höchstens T Schritten, welche $\bar{y}$ ausgibt.*

Beweis: Der Beweis läuft analog zum Beweis des vorherigen Satzes. Der einzige Unterschied im Beweis zu *(i)* besteht darin, daß der Fall eintreten kann, daß der Knoten η mit $x_i \leftarrow M$ gelabelt ist. In diesem Fall wird der Zeiger des zweiten Bandes i Schritte nach rechts bewegt, dann mit dem Befehl `(guess 2)` ein Element aus M geraten, der Zeiger wieder um i Schritte nach links bewegt, der Zähler für die durchlaufenen Blöcke um eins erhöht, und das Programm springt in die Anfangszeile des zum Knoten $\beta(\eta)$ gehörigen Blockes.

Im Beweis zu *(ii)* besteht der Unterschied darin, daß wir für einen Befehl $\Pi_q =$`(guess` i`)` mit $i \in [k]$ die Subroutine $\texttt{ptrpos}_k(i)$, einen mit $x_{3i} \leftarrow M$ gelabelten Knoten und einen mit $x_{3i-2} \leftarrow 1$ gelabelten Knoten hinzufügen. Die ausgehende Kante endet wieder am Knoten η_{q+1} (siehe Abbildung 3.7).

Die übrigen Schritte der Konstruktion von $\mathbb{A}_{\mathbb{F}}$ und $\mathbb{F}_{\mathbb{A}}$ werden wie im binär nichtdeterministischen Fall durchgeführt. Die Anzahl der benötigten Schritte beziehungsweise Zustandswechsel ist dieselbe wie im binär nichtdeterminstischen Fall. □

Kapitel 4

Die polynomiellen Hierarchien

In diesem Kapitel werden wir die Definitionen der Klassen $\mathrm{NP}_{\mathcal{M}}$ und $\mathrm{BNP}_{\mathcal{M}}$ verallgemeinern, was uns auf die Begriffe der polynomiellen Hierarchie und der nach einer Menge $R \subseteq M$ relativierten polynomiellen Hierarchie im Sinne einer Struktur $\mathcal{M}$ führen wird. Anschließend betrachten wir Probleme, die eine bestimmte Komplexitätsklasse in gewissem Sinne charakterisieren, also für die entsprechende Klasse vollständige Probleme. Die in diesem Abschnitt gezeigten Ergebnisse sind im wesentlichen die in [22] präsentierten Ergebnisse. Da wir immer auch an den klassischen Fragen, ob die polynomielle Hierarchie im Sinne der Standardstruktur auf die i-te Ebene kollabiert, interessiert sind, stellen wir in den H_i-Standardtypen Eigenschaften von Strukturen zusammen, die gewährleisten, daß sich die betreffende Struktur bezüglich dieser Frage genauso verhält, wie die Standardstruktur. Der Begriff des H_i-Standardtyps und die mit diesem zusammenhängenden Ergebnisse sind Verallgemeinerungen des in [22] entwickelten Begriffs des Standardtyps bezüglich der P = NP-Frage und der entsprechenden Ergebnisse für diesen Begriff. Abschließend zeigen wir ein Transfertheorem für elementar äquivalente Strukturen. Dieses wurde in [16] mit Hilfe von Berechnungsbäumen bewiesen. Unser Beweis wird auf einer von Moritz Müller durchgeführten Verallgemeinerung des Beweises des Transfertheorems für algebraisch abgeschlossene Körper der Charakteristik 0 aus [3] beruhen und ohne Berechnungsbäume auskommen.

4.1 $\mathrm{PH}_{\mathcal{M}}$ und $\mathrm{RPH}_{\mathcal{M}}$

Wir haben in Satz 2.2.16 gesehen, daß für eine Struktur $\mathcal{M}$ gerade diejenigen Probleme X in der Klasse $\mathrm{NP}_{\mathcal{M}}$ liegen, deren Ja-Instanzen bei Vorlage eines geeigneten Zeugens schnell verifiziert werden können. Für alle $\bar{x} \in M^*$ gilt also

$$\bar{x} \in X \Leftrightarrow \text{ es gibt ein } \bar{y} \in M^{p(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in Y,$$

wobei $p(n)$ ein geeignetes Polynom und Y ein entsprechendes Problem aus $\mathrm{P}_{\mathcal{M}}$ ist. Wir werden nun diesen Ansatz verallgemeinern, und somit die Klassen der polynomiellen Hierarchie über einer Struktur $\mathcal{M}$ erhalten. Zunächst führen wir noch die folgende Schreibweise ein:

Schreibweise 4.1.1 Für eine Komplexitätsklasse $\mathcal{C}$ von Problemen aus M^* bezeichnen wir mit $co\mathcal{C}$ die Klasse der Komplemente der Probleme aus $\mathcal{C}$:

$$co\mathcal{C} := \{M^* \setminus X \mid X \in \mathcal{C}\}.$$

Bemerkung 4.1.2 Wenn $\mathcal{C}_1 \subseteq \mathcal{C}_2$, so gilt auch $co\mathcal{C}_1 \subseteq co\mathcal{C}_2$.

Hiermit können wir nun die polynomielle Hierarchie im Sinne einer Struktur $\mathcal{M}$ definieren:

Definition 4.1.3 ($\Sigma_i\mathrm{P}_\mathcal{M}$ und $\Pi_i\mathrm{P}_\mathcal{M}$ und $\mathrm{PH}_\mathcal{M}$) Wir definieren die *polynomielle Hierarchie im Sinne einer Struktur* $\mathcal{M}$ induktiv:

- $\Sigma_0\mathrm{P}_\mathcal{M} := \mathrm{P}_\mathcal{M}$
- $\Pi_i\mathrm{P}_\mathcal{M} := co\Sigma_i\mathrm{P}_\mathcal{M}$
- $\Delta_i\mathrm{P}_\mathcal{M} := \Sigma_i\mathrm{P}_\mathcal{M} \cap \Pi_i\mathrm{P}_\mathcal{M}$
- Ein Problem $X \subseteq M^*$ ist in der Klasse $\Sigma_{i+1}\mathrm{P}_\mathcal{M}$, falls es ein Polynom $p(n)$ und ein Problem $Y \in \Pi_i\mathrm{P}_\mathcal{M}$ gibt, so daß für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \text{ es gibt ein } \bar{y} \in M^{p(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in Y.$$

- $\mathrm{PH}_\mathcal{M} := \bigcup_{i\in\mathbb{N}} \Sigma_i\mathrm{P}_\mathcal{M}$

Für $i \in \mathbb{N}_0$ sagen wir, daß *die polynomielle Hierarchie im Sinne von* $\mathcal{M}$ *auf die* i*-te Ebene kollabiert*, falls für alle $j \in \mathbb{N}$ mit $j > i$ gilt:

$$\Sigma_j\mathrm{P}_\mathcal{M} = \Delta_j\mathrm{P}_\mathcal{M} = \Pi_j\mathrm{P}_\mathcal{M} = \Sigma_i\mathrm{P}_\mathcal{M}.$$

Analog lässt sich für eine Menge $R \subseteq M$ durch Verallgemeinern aus der Klasse $\mathrm{BNP}_\mathcal{M}$ die nach R relativierte polynomielle Hierarchie im Sinne von $\mathcal{M}$ konstruieren:

Definition 4.1.4 ($\mathrm{R}\Sigma_i\mathrm{P}_\mathcal{M}$ und $\mathrm{R}\Pi_i\mathrm{P}_\mathcal{M}$ und $\mathrm{RPH}_\mathcal{M}$) Für eine Struktur $\mathcal{M}$ und eine Menge $R \subseteq M$ definieren wir die *nach* R *relativierte polynomielle Hierarchie im Sinne von* $\mathcal{M}$ induktiv wie folgt:

- $\mathrm{R}\Sigma_0\mathrm{P}_\mathcal{M} := \mathrm{P}_\mathcal{M}$
- $\mathrm{R}\Pi_i\mathrm{P}_\mathcal{M} := co\mathrm{R}\Sigma_i\mathrm{P}_\mathcal{M}$
- $\mathrm{R}\Delta_i\mathrm{P}_\mathcal{M} := \mathrm{R}\Sigma_i\mathrm{P}_\mathcal{M} \cap \mathrm{R}\Pi_i\mathrm{P}_\mathcal{M}$
- Ein Problem $X \subseteq M^*$ ist in der Klasse $\mathrm{R}\Sigma_{i+1}\mathrm{P}_\mathcal{M}$, falls es ein Polynom $p(n)$ und ein Problem $Y \in \mathrm{R}\Pi_i\mathrm{P}_\mathcal{M}$ gibt, so daß für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \text{ es gibt ein } \bar{y} \in R^{p(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in Y.$$

- $\mathrm{RPH}_\mathcal{M} := \bigcup_{i\in\mathbb{N}} \mathrm{R}\Sigma_i\mathrm{P}_\mathcal{M}$

Die nach $\{0,1\}$ relativierte Hierarchie im Sinne von $\mathcal{M}$ nennen wir auch die *boolesch relativierte Hierarchie im Sinne von* $\mathcal{M}$ und schreiben für $\{0,1\}\mathrm{PH}_{\mathcal{M}}$ auch $\mathrm{BPH}_{\mathcal{M}}$, für $\{0,1\}\Sigma_i\mathrm{P}_{\mathcal{M}}$ auch $\mathrm{B}\Sigma_i\mathrm{P}_{\mathcal{M}}$ und für $\{0,1\}\Pi_i\mathrm{P}_{\mathcal{M}}$ auch $\mathrm{B}\Pi_i\mathrm{P}_{\mathcal{M}}$.

Analog werden die nicht uniforme polynomielle Hierarchie $\mathbb{PH}_{\mathcal{M}}$, die nicht gesetzte nicht uniforme polynomielle Hierarchie $\mathbb{PH}^*_{\mathcal{M}}$ und die entsprechenden nach R relativierten Hierarchien $\mathrm{R}\mathbb{PH}_{\mathcal{M}}$ und $\mathrm{R}\mathbb{PH}^*_{\mathcal{M}}$ über einer Struktur $\mathcal{M}$ definiert.

Bemerkung 4.1.5 **($\mathrm{PH}_{\mathcal{M}} = \mathbf{M}\mathrm{PH}_{\mathcal{M}}$)** Die nichtrelativierte polynomielle Hierarchie $\mathrm{PH}_{\mathcal{M}}$ ist eigentlich ein Spezialfall der relativierten polynomiellen Hierarchie $\mathrm{RPH}_{\mathcal{M}}$. Setzt man $R = M$ in der Definition der relativierten Hierarchie, so erhält man gerade die nichtrelativierte Hierarchie.

Die Klassen der Hierarchien haben dann eine bestimmte Darstellung:

Bemerkung 4.1.6 Für ein Problem $X \in \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$ gibt es dann Polynome $p_1(n)\ldots,p_i(n)$ und ein Problem $Y \in \mathrm{P}_{\mathcal{M}}$, so daß (in abkürzender Schreibweise) für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \exists \bar{y_1} \in R^{p_1(|\bar{x}|)} \forall \bar{y_2} \in R^{p_2(|\bar{x}|)} \ldots Q_i \bar{y_i} \in R^{p_i(|\bar{x}|)} \text{ mit } \bar{x}\bar{y_1}\bar{y_2}\ldots\bar{y_i} \in Y$$

Der Quantor Q_i ist dabei ein Existenzquantor, falls i ungerade ist, andernfalls ein Allquantor.

Wenn das Problem X andererseits aus der Klasse $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}}$ ist, so gibt es Polynome $p_1(n)\ldots,p_i(n)$ und ein Problem $Y \in \mathrm{P}_{\mathcal{M}}$, so daß (in abkürzender Schreibweise) für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \forall \bar{y_1} \in R^{p_1(|\bar{x}|)} \exists \bar{y_2} \in R^{p_2(|\bar{x}|)} \ldots Q_i \bar{y_i} \in R^{p_i(|\bar{x}|)} \text{ mit } \bar{x}\bar{y_1}\bar{y_2}\ldots\bar{y_i} \in Y$$

Hierbei ist der Quantor Q_i ein Allquantor, falls i ungerade ist, ein Existenzquantor sonst.

Bemerkung 4.1.7 **($\mathrm{NP}_{\mathcal{M}} = \Sigma_1\mathrm{P}_{\mathcal{M}}$)** Nach Definition gilt $\Sigma_1\mathrm{P}_{\mathcal{M}} = \mathrm{VP}_{\mathcal{M}}$ und mit Satz 2.2.16 gilt damit $\mathrm{NP}_{\mathcal{M}} = \Sigma_1\mathrm{P}_{\mathcal{M}}$.

Bemerkung 4.1.8 Sind zwei Strukturen $\mathcal{M}$ und $\mathcal{N}$ isomorph via I, und ist ein Problem $X \subseteq M^*$ für eine natürliche Zahl i aus der Klasse $\Sigma_i\mathrm{P}_{\mathcal{M}}$ oder $\Pi_i\mathrm{P}_{\mathcal{M}}$, so ist das Problem $I(X) = \{\bar{x} \in N^* \mid I^{-1}(x_1)\ldots I^{-1}(x_{|\bar{x}|}) \in X\}$ in der entsprechenden Klasse $\Sigma_i\mathrm{P}_{\mathcal{N}}$ beziehungsweise $\Pi_i\mathrm{P}_{\mathcal{N}}$.

Falls die Menge, nach der relativiert wird, schnell entscheidbar ist, so sind die relativierten Klassen in den entsprechenden nicht relativierten Klassen enthalten:

Lemma 4.1.9 *Sei $\mathcal{M}$ eine Struktur und $R \subseteq M$ eine Teilmenge des Universums von $\mathcal{M}$, so daß $R \in \mathrm{P}_{\mathcal{M}}$ gilt. Dann gilt für jedes $i \in \mathbb{N}$:*

$$\mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}} \subseteq \Sigma_i\mathrm{P}_{\mathcal{M}}$$

und somit auch $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}} \subseteq \Pi_i\mathrm{P}_{\mathcal{M}}$.

Beweis: Sei also $X \in \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$. Dann gibt es Polynome $p_1(n), \ldots, p_i(n)$ und ein Problem $Y \in \mathrm{P}_{\mathcal{M}}$, und für alle $\bar{x} \in M^*$ gilt:

$$\begin{aligned} \bar{x} \in X &\Leftrightarrow \exists \bar{y_1} \in R^{p_1(|\bar{x}|)} \ldots Q_i \bar{y_i} \in R^{p_i(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in Y \\ &\Leftrightarrow \exists \bar{y_1} \in M^{p_1(|\bar{x}|)} \ldots Q_i \bar{y_i} \in M^{p_i(|\bar{x}|)} \text{ mit } \bar{y} \in R^* \text{ und } \bar{x}\bar{y} \in Y. \end{aligned}$$

Die Bedingung „$\bar{y} \in R^*$ und $\bar{y}\bar{x} \in Y$ ist nun, da R in $\mathrm{P}_{\mathcal{M}}$ ist, in polynomieller Zeit im Sinne von $\mathcal{M}$ entscheidbar. Somit gilt $X \in \Sigma_i\mathrm{P}_{\mathcal{M}}$. Da $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}} = co\mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$ und $\Pi_i\mathrm{P}_{\mathcal{M}} = co\Sigma_i\mathrm{P}_{\mathcal{M}}$, gilt damit auch $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}} \subseteq \Pi_i\mathrm{P}_{\mathcal{M}}$. □

Korollar 4.1.10 *Für jede Struktur $\mathcal{M}$ und für jedes $i \in \mathbb{N}$ gilt $\mathrm{B}\Sigma_i\mathrm{P}_{\mathcal{M}} \subseteq \Sigma_i\mathrm{P}_{\mathcal{M}}$ und $\mathrm{B}\Pi_i\mathrm{P}_{\mathcal{M}} \subseteq \Pi_i\mathrm{P}_{\mathcal{M}}$.* □

Eine große unbeantwortete Frage im Klassischen ist die Frage, ob die polynomielle Hierarchie auf irgendeine Ebene kollabiert, ob also $\mathrm{PH} = \Sigma_i\mathrm{P}$ für irgendeine natürliche Zahl i gilt. Um zu prüfen, ob die polynomielle Hierarchie im Sinne einer Struktur auf die i-te Ebene kollabiert, reicht es dabei, wie wir sehen werden, zu prüfen, ob die Klasse $\Sigma_i\mathrm{P}_{\mathcal{M}}$ unter Komplementbildung abgeschlossen ist. Allgemeiner gilt im relativierten Fall:

Lemma 4.1.11 *Sei $\mathcal{M}$ eine Struktur, sei $R \subseteq M$ und sei $i \in \mathbb{N}_{>0}$. Falls $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}} = \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$, so kollabiert die nach R relativierte polynomielle Hierarchie im Sinne von $\mathcal{M}$ auf die i-te Ebene.*

Beweis: Sei also $\mathcal{M}$ eine Struktur und gelte $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}} \subseteq \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$ für ein $i \in \mathbb{N}_{>0}$. Da $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}}$ gerade die Komplemente der Probleme aus $\mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$ enthält, gilt damit auch $\mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}} \subseteq \mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}}$ und somit $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}} = \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$. Es genügt die Inklusion

$$\mathrm{R}\Sigma_{i+1}\mathrm{P}_{\mathcal{M}} \subseteq \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$$

zu zeigen. Die allgemeinere Aussage, daß die Inklusion

$$\mathrm{R}\Sigma_{i+l}\mathrm{P}_{\mathcal{M}} \subseteq \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$$

für alle $l \in \mathbb{N}$ gilt, folgt dann mit Induktion.

Betrachten wir also ein beliebiges Problem $X \in \mathrm{R}\Sigma_{i+1}\mathrm{P}_{\mathcal{M}}$. Nach Definition gibt es ein Polynom $p(n)$ und ein Problem $X' \in \mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}}$, so daß für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \text{ es gibt ein } \bar{y} \in R^{p(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in X'.$$

Nach Vorraussetzung gilt $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}} \subseteq \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$ und somit auch $X' \in \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$. Es gibt also ein Polynom $p'(n)$ und ein Problem $X'' \in \mathrm{R}\Pi_{i-1}\mathrm{P}_{\mathcal{M}}$, so daß gilt:

$$\bar{x}\bar{y} \in X' \Leftrightarrow \text{ es gibt ein } \bar{z} \in R^{p'(|\bar{x}|)} \text{ mit } \bar{x}\bar{y}\bar{z} \in X''.$$

Insgesamt gilt also für alle $\bar{x} \in M^*$:

$$\bar{x} \in X \Leftrightarrow \text{ es gibt ein } \bar{y'} \in R^{p(|\bar{x}|)+p'(|\bar{x}|)} \text{ mit } \bar{x}\bar{y'} \in X''$$

und somit ist das Problem X in der Klasse $\mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$. □

Da man ein Problem der Klasse $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}}$ durch Quantifizieren „blinder“ Variablen, welche in der zu prüfenden Relation keine Rolle spielen, ohne weiteres zu einem Problem aus $\mathrm{R}\Sigma_{i+1}\mathrm{P}_{\mathcal{M}}$ machen kann, ergibt sich aus dem Lemma das folgende Korollar.

Korollar 4.1.12 *Sei $\mathcal{M}$ eine Struktur und sei $R \subseteq M$. Für alle $i, j \in \mathbb{N}$ mit $i < j$ gilt:*

$$\mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}} = \mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}} \Leftrightarrow \mathrm{RPH}_{\mathcal{M}} = \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}.$$

Beweis: „←“: Wenn $\mathrm{RPH}_{\mathcal{M}} = \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$ gilt, so gilt nach Definition $\mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}} = \mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}}$.

„→“: Da $\mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}} \subseteq \mathrm{R}\Pi_j\mathrm{P}_{\mathcal{M}}$ und nach Voraussetzung $\mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}} = \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$, gilt $\mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}} \subseteq \mathrm{R}\Pi_j\mathrm{P}_{\mathcal{M}}$. Da die Klasse $\mathrm{R}\Pi_j\mathrm{P}_{\mathcal{M}}$ gerade die Komplemente von $\mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}}$ enthält, gilt damit $\mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}} = \mathrm{R}\Pi_j\mathrm{P}_{\mathcal{M}}$ und die nach R relativierte polynomielle Hierarchie im Sinne von $\mathcal{M}$ kollabiert auf die j-te Ebene. Da nach Voraussetzung $\mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}} = \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$ kollabiert sie also auf die i-te Ebene. □

Mit $R = M$ folgen die letzten beiden Aussagen sofort für die nichtrelativierten Hierarchien:

Korollar 4.1.13 *Sei $\mathcal{M}$ eine Struktur und seien $i, j \in \mathbb{N}$ mit $i < j$. Dann gilt:*

(i) Falls $\Pi_j\mathrm{P}_{\mathcal{M}} = \Sigma_j\mathrm{P}_{\mathcal{M}}$, so kollabiert die polynomielle Hierarchie im Sinne von $\mathcal{M}$ auf die j-te Ebene.

(ii) Die polynomielle Hierarchie im Sinne von $\mathcal{M}$ kollabiert genau dann auf die i-te Ebene, wenn $\Sigma_i\mathrm{P}_{\mathcal{M}} = \Sigma_j\mathrm{P}_{\mathcal{M}}$ gilt.

□

Bemerkung 4.1.14 Die Frage, ob in einer Struktur $\mathcal{M}$ die Gleichheit der Klassen $\mathrm{P}_{\mathcal{M}}$ und $\mathrm{NP}_{\mathcal{M}}$ gilt, ist nach diesen Überlegungen äquivalent zu der Frage, ob die polynomielle Hierarchie im Sinne von $\mathcal{M}$ auf die 0-te Ebene kollabiert, es gilt:

$$\mathrm{P}_{\mathcal{M}} = \mathrm{NP}_{\mathcal{M}} \Leftrightarrow \mathrm{PH}_{\mathcal{M}} = \Sigma_0\mathrm{P}_{\mathcal{M}}.$$

4.2 Reduktionen und Vollständigkeit

Wie im klassischen Fall gibt es auch bei dem auf beliebige Strukturen verallgemeinerten Berechenbarkeitsbegriff gewisse, für die verschiedenen Komplexitätsklassen „charakteristische“ Probleme. Dies bedeutet, daß man jedes Problem einer Komplexitätsklasse auf ein solches für diese Klasse charakteristisches Problem schnell zurückführen kann. Wenn man dann einen Algorithmus für das charakteristische Problem hat, so kann man jedes andere Problem dieser Klasse in nur unwesentlich längerer Zeit lösen.

Die Idee der schnellen Rückführbarkeit fassen wir für uniforme Komplexitätsklassen in folgende Definition:

Definition 4.2.1 ($\mathrm{P}_{\mathcal{M}}$-Reduktion) Sei $\mathcal{M}$ eine Struktur und seien $X \subseteq M^*$ und $Y \subseteq M^*$ zwei Probleme über dieser Struktur. Eine *$\mathrm{P}_{\mathcal{M}}$-Reduktion von X auf Y* ist eine in polynomieller Zeit berechenbare Funktion $f : M^* \to M^*$, so daß für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \Leftrightarrow f(\bar{x}) \in Y$$

Das Problem X heißt, falls es eine solche $\mathrm{P}_{\mathcal{M}}$-Reduktion von X auf Y gibt, *$\mathrm{P}_{\mathcal{M}}$-reduzierbar auf Y*. Wir schreiben dann: $X \leq^{\mathrm{P}_{\mathcal{M}}} Y$.

Da sich uniforme und nicht uniforme Berechnungen manchmal deutlich unterscheiden, definieren wir für den nicht uniformen Fall die $\mathbb{P}_{\mathcal{M}}$-Reduktion:

Definition 4.2.2 ($\mathbb{P}_{\mathcal{M}}$-Reduktion) Für eine Struktur $\mathcal{M}$ und zwei Probleme X und Y über $\mathcal{M}$ ist eine *$\mathbb{P}_{\mathcal{M}}$-Reduktion von X auf Y* eine Funktion $f : M^* \to M^*$, so daß gilt:

- Es gibt eine gesetzte Familie $(C_n)_{n\in\mathbb{N}_{>0}}$ von Schaltkreisen polynomieller Größe über $\mathcal{M}$, so daß für alle $\bar{x} \in M^*$ gilt: $C_{|\bar{x}|}(\bar{x}) = f(\bar{x})$
- Für alle $\bar{x} \in M^*$ gilt: $\bar{x} \in X \Leftrightarrow f(\bar{x}) \in Y$

Das Problem X heißt, falls eine solche Funktion existiert, *$\mathbb{P}_{\mathcal{M}}$-reduzierbar auf Y*, und wir schreiben: $X \leq^{\mathbb{P}_{\mathcal{M}}} Y$.

Wenn ein Problem für eine bestimmte Komplexitätsklasse charakteristisch ist, sollte also jedes andere Problem dieser Klasse auf es reduzierbar sein. Zusätzlich fordern wir noch, daß das Problem selber auch in der Klasse enthalten ist, für welche es charakteristisch ist.

Definition 4.2.3 (Vollständigkeit) Sei $\mathcal{M}$ eine Struktur. Sei $R \subseteq M$ und $\mathcal{C}$ eine Komplexitätsklasse aus der uniformen nach R relativierten polynomiellen Hierarchie im Sinne von $\mathcal{M}$. Ein Problem $X \subseteq M^*$ heißt *$\mathcal{C}$-vollständig bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion*, falls $X \in \mathcal{C}$ und für jedes andere Problem $Y \in \mathcal{C}$ gilt: $Y \leq^{\mathrm{P}_{\mathcal{M}}} X$.

Für eine nicht uniforme Klasse $\mathcal{C}$ aus $\mathrm{R}\mathbb{P}\mathbb{H}_{\mathcal{M}}$ ist das Problem X analog *$\mathcal{C}$-vollständig bezüglich $\mathbb{P}_{\mathcal{M}}$-Reduktion*, falls $X \in \mathcal{C}$ und für alle $Y \in \mathcal{C}$ gilt: $Y \leq^{\mathbb{P}_{\mathcal{M}}} X$.

Das Resultat, daß ein für eine bestimmte Klasse vollständiges Problem auch wirklich chararkteristisch für diese Klasse ist, liefert das folgende Lemma:

Lemma 4.2.4 *Sei $\mathcal{M}$ eine Struktur. Sei $R \subseteq M$ und i eine natürliche Zahl. Sei weiterhin $\mathcal{C}$ eine Komplexitätsklasse mit $\mathcal{C} = \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}}$ oder $\mathcal{C} = \mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}}$ und sei X ein Problem aus $\mathcal{C}$. Dann gilt für jedes Problem $Y \subseteq M^*$:*

$$Y \leq^{\mathrm{P}_{\mathcal{M}}} X \Rightarrow Y \in \mathcal{C}.$$

In den nicht uniformen Fällen analog.

Beweis: Sei also $X \in \mathcal{C}$ und $Y \subseteq M^*$ mit $Y \leq^{\mathrm{P}_{\mathcal{M}}} X$. Es gibt also eine in polynomieller Zeit im Sinne von $\mathcal{M}$ berechenbare Funktion $f : M^* \to M^*$, so daß für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in Y \Leftrightarrow f(\bar{x}) \in X.$$

Da $X \in \mathcal{C}$ gibt es andererseits Polynome $p_1(n), \ldots, p_i(n)$ und ein $X' \in \mathrm{P}_{\mathcal{M}}$, so daß für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \Leftrightarrow Q_1\bar{y_1} \in R^{p_1(|\bar{x}|)} \ldots Q_i\bar{y_i} \in R^{p_1(|\bar{x}|)} \text{ mit } \bar{x}\bar{y_1}\ldots\bar{y_i} \in X'.$$

Insgesamt gilt also für alle $\bar{x} \in M^*$:

$$\bar{x} \in Y \Leftrightarrow Q_1\bar{y_1} \in R^{p_1(|\bar{x}|)} \ldots Q_i\bar{y_i} \in R^{p_1(|\bar{x}|)} \text{ mit } f(\bar{x})\bar{y_1}\ldots\bar{y_i} \in X'.$$

Da die Funktion f in polynomieller Zeit berechenbar und X' aus $\mathrm{P}_{\mathcal{M}}$ ist, ist auch das Problem $Y' := \{\bar{x}\bar{y}_1 \dots \bar{y}_i \in M^* \mid f(\bar{x})\bar{y}_1 \dots \bar{y}_i \in X'\}$ in $\mathrm{P}_{\mathcal{M}}$. Somit ist aber Y in der Klasse $\mathcal{C}$.

In den nicht uniformen Fällen analog. □

Hiermit erhalten wir das für die Untersuchungen der polynomiellen Hierarchien notwendige Resultat:

Korollar 4.2.5 *Sei $\mathcal{M}$ eine Struktur und $R \subseteq M$. Seien $\mathcal{C}_1, \mathcal{C}_2$ Komplexitätsklassen, so daß es natürliche Zahlen i_1 und i_2 gibt mit $\mathcal{C}_j = \mathrm{R}\Sigma_{i_j}\mathrm{P}_{\mathcal{M}}$ oder $\mathcal{C}_j = \mathrm{R}\Pi_{i_j}\mathrm{P}_{\mathcal{M}}$ für $i \in \{1, 2\}$, und sei X ein bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\mathcal{C}_1$-vollständiges Problem. Dann gilt:*

$$X \in \mathcal{C}_2 \Rightarrow \mathcal{C}_1 \subseteq \mathcal{C}_2.$$

In den nicht uniformen Fällen analog.

Beweis: Sei X also $\mathcal{C}_1$-vollständig bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion und $Y \in \mathcal{C}_1$ beliebig. Nach Definition von $\mathcal{C}$-Vollständigkeit gilt also $Y \leq^{\mathrm{P}_{\mathcal{M}}} X$. Mit der Voraussetzung $X \in \mathcal{C}_2$ und dem vorangegangenen Lemma also auch $Y \in \mathcal{C}_2$.

In den nicht uniformen Fällen analog. □

Bemerkung 4.2.6 Sei $\mathcal{M}$ eine Struktur. Wenn ein Problem X für eine Komplexitätsklasse $\mathcal{C} \in \mathrm{PH}_{\mathcal{M}}$ bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion vollständig ist, so ist $M^* \setminus X$ für die Klasse $co\mathcal{C}$ vollständig bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion.

Betrachten wir nun die Klasse $\mathrm{NP}_{\mathcal{M}} = \Sigma_1\mathrm{P}_{\mathcal{M}}$. Um ein bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\mathrm{NP}_{\mathcal{M}}$-vollständiges Problem X zu finden, stellen wir folgende Überlegung an:

Für ein Problem $Y \in \mathrm{NP}_{\mathcal{M}}$ lassen sich nach Satz 2.2.16 die „Ja"-Instanzen zusammen mit einem passenden Zeugen in polynomieller Zeit verifizieren. Es gilt somit $\{\bar{x}\bar{y} \mid \bar{x} \in Y \text{ und } \bar{y} \text{ ist Zeuge für } \bar{x}\} \in \mathrm{P}_{\mathcal{M}}$. Nach Satz 2.1.13 gibt es dann eine in polynomieller Zeit berechenbare gesetzte Schaltkreisfamilie $(C_n)_{n\in\mathbb{N}_{>0}}$, welche diese Menge akzeptiert. Wenn wir also bei gegebenem $\bar{x} \in M^*$ wissen, ob es einen Zeugen gibt, so daß der entsprechende Schaltkreis das Element $\bar{x}$ zusammen mit dem Zeugen akzeptiert, so wissen wir sofort, daß $\bar{x} \in X$ gilt. Wir betrachten also folgendes Problem:

SAT(SK MIT PAR IN $\mathcal{M}$)
Eingabe: Ein Schaltkreis $C(\bar{x}, \bar{y})$ in binärer Kodierung, $\bar{a} \in M^{\lvert\bar{x}\rvert}$
Frage: Gibt es ein $\bar{b} \in M^{\lvert\bar{y}\rvert}$ mit $C(\bar{a}, \bar{b}) = 1$?

Und in der Tat gilt:

Satz 4.2.7 *Das Problem* SAT(SK MIT PAR IN $\mathcal{M}$) *ist $\mathrm{NP}_{\mathcal{M}}$-vollständig bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion und $\mathbb{NP}_{\mathcal{M}}$-vollständig bezüglich $\mathbb{P}_{\mathcal{M}}$-Reduktion für alle Strukturen $\mathcal{M}$.*

Beweis: Wir müssen zeigen, daß SAT(SK MIT PAR IN $\mathcal{M}$) in den Klassen $\mathrm{NP}_{\mathcal{M}}$und $\mathbb{NP}_{\mathcal{M}}$liegt, und daß jedes andere Problem aus diesen Klassen $\mathrm{P}_{\mathcal{M}}$- beziehungsweise $\mathbb{P}_{\mathcal{M}}$-reduzierbar auf SAT(SK MIT PAR IN $\mathcal{M}$) ist.

Um die Zugehörigkeit zu $\mathrm{NP}_{\mathcal{M}}$zu zeigen, konstruieren wir wie folgt eine vollständig nichtdeterministische Turingmaschine $\mathbb{A}$ über $\mathcal{M}$, welche das Problem Sat(Sk mit Par in $\mathcal{M}$) in polynomieller Zeit akzeptiert:
Bei Eingabe von $(\ulcorner C(\bar{x},\bar{y})\urcorner,\bar{a})$ berechnet $\mathbb{A}$ zunächst $|\bar{y}|$ und rät ein $\bar{b}\in M^{|\bar{y}|}$. Dann wertet sie $C(\bar{a},\bar{b})$ mit Proposition 2.1.6 in polynomieller Zeit aus und gibt 1 aus, falls $C(\bar{a},\bar{b})=1$, andernfalls 0.

Da nach Korollar 2.2.17 die Inklusion $\mathrm{NP}_{\mathcal{M}}\subseteq\mathbb{NP}_{\mathcal{M}}$ gilt, ist Sat(Sk mit Par in $\mathcal{M}$) somit auch in der Klasse $\mathbb{NP}_{\mathcal{M}}$.

Für die $\mathrm{NP}_{\mathcal{M}}$-Vollständigkeit bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion betrachten wir ein beliebiges Problem $X\in\mathrm{NP}_{\mathcal{M}}$. Nach Satz 2.2.16 gibt es ein $Y\in\mathrm{P}_{\mathcal{M}}$ und ein Polynom q, so daß für alle $\bar{x}\in M^*$ gilt: $\bar{x}\in X\Leftrightarrow$ es gibt ein $\bar{y}\in M^{q(|\bar{x}|)}$ mit $\bar{x}\bar{y}\in Y$. Da $Y\in\mathrm{P}_{\mathcal{M}}$ gibt es nach Satz 2.1.13 dann eine Familie $(C_n)_{n\in\mathbb{N}_{>0}}$ von gesetzten Schaltkreisen, die das Problem Y akzeptiert, so daß die Funktion $1^n\mapsto\ulcorner C_n\urcorner$ in polynomieller Zeit berechenbar ist. Eine Maschine $\mathbb{A}$, welche das Problem X auf Sat(Sk mit Par in $\mathcal{M}$) reduziert, geht nun folgendermaßen vor:
Bei Eingabe von $\bar{a}\in M^*$ berechnet $\mathbb{A}$ (nach obigen Überlegungen in polynomieller Zeit) die Kodierung des Schaltkreises $C_{|\bar{a}|+q(|\bar{a}|)}$. Die binäre Kodierung des entsprechenden konstantenfreien Schaltkreises $C'_{|\bar{a}|+q(|\bar{a}|)}(\bar{w},\bar{x},\bar{y})$ zusammen mit den von $C_{|\bar{a}|+q(|\bar{a}|)}$ verwendeten Konstanten $\bar{c}$ und dem Element $\bar{a}$ ist eine Instanz von Sat(Sk mit Par in $\mathcal{M}$), und es gibt genau dann ein $\bar{b}\in M^{q(|\bar{a}|)}$ mit $C'_{|\bar{a}|+q(|\bar{a}|)}(\bar{c},\bar{a},\bar{b})=1$ und somit mit $C(\bar{a},\bar{b})=1$, wenn $\bar{a}\in X$ gilt.

Die $\mathbb{NP}_{\mathcal{M}}$-Vollständigkeit bezüglich $\mathbb{P}_{\mathcal{M}}$-Reduktion läßt sich analog beweisen, indem man einfach die Definitionen von $\mathbb{NP}_{\mathcal{M}}$ und $\mathbb{P}_{\mathcal{M}}$ verwendet. □

Bemerkung 4.2.8 Analog gilt für jede Struktur $\mathcal{M}$, daß das Problem Σ_iSat(SK mit Par in $\mathcal{M}$) bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\Sigma_i\mathrm{P}_{\mathcal{M}}$-vollständig ist. Hierbei ist Σ_iSat(SK mit Par in $\mathcal{M}$) das Problem, bei Eingabe eines Schaltkreises $C(\bar{x},\bar{y_1},\ldots,\bar{y_i})$ in binärer Kodierung und Parametern $\bar{a}\in M^{|\bar{x}|}$ zu entscheiden, ob gilt:

$$\exists\bar{b_1}\in M^{|\bar{y_1}|}\forall\bar{b_2}\in M^{|\bar{y_2}|}\ldots Q_i\bar{b_i}\in M^{|\bar{y_i}|}\text{ mit }C(\bar{a},\bar{b_1},\ldots,\bar{b_i})=1.$$

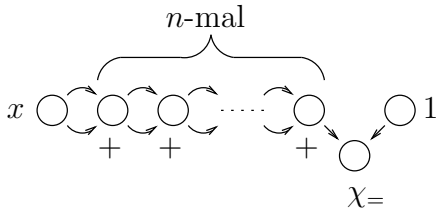
Hierbei ist Q_i wieder ein Existenzquantor, falls i ungerade ist, ein Allquantor sonst.

Die Frage, ob es für einen Schaltkreis $C(\bar{x},\bar{y})$ und ein $\bar{a}\in M^*$ ein passendes $\bar{b}\in M^*$ gibt, so daß $C(\bar{a},\bar{b})=1$ gilt, lässt sich auch auf eine andere Weise interpretieren. Eine Sichtweise auf einen solchen Schaltkreis C ist, daß er eine quantorenfreie Formel φ_C repräsentiert. Die Frage, ob es für C und $\bar{a}$ ein $\bar{b}$ mit $C(\bar{a},\bar{b})=1$ gibt, ist dann gleichbedeutend mit der Frage, ob es ein $\bar{b}$ gibt, so daß die Struktur $\mathcal{M}$ Modell von $\varphi_C(\bar{a},\bar{b})$ ist. Dies motiviert das nächste Problem.

Sat(Qf Fml mit Par in $\mathcal{M}$)
Eingabe: Eine quantorenfreie Formel $\varphi(\bar{x},\bar{y})$ in binärer Kodierung, $\bar{a}\in M^{\lvert\bar{x}\rvert}$
Frage: Gibt es ein $\bar{b}\in M^{\lvert\bar{y}\rvert}$ mit $\mathcal{M}\models\varphi(\bar{a},\bar{b})$?

Eine Schwierigkeit für eine $\mathrm{P}_{\mathcal{M}}$- oder $\mathbb{P}_{\mathcal{M}}$-Reduktion ist hierbei allerdings, daß die durch einen Schaltkreis repräsentierte Formel im Vergleich zum Schaltkreis gewissermaßen zu

groß sein kann. Dies liegt daran, daß in einem Schaltkreis von einem Knoten mehrere Pfeile ausgehen können. Betrachten wir zum Beispiel die Struktur $\mathcal{R} = (\mathbb{R}, +, P)$ mit der für eine positive natürliche Zahl n durch $P := \{x \in \mathbb{R} \mid 2^n x = 1\}$ definierten einstelligen Relation P und den Schaltkreis



welcher gerade die Menge $\{x \mid x \in P\}$ akzeptiert. Dieser Schaltkreis repräsentiert die quantorenfreie Formel

$$\overbrace{x + x + \ldots + x}^{2^n\text{-mal}} = 1.$$

Die Länge dieser Formel ist exponentiell in der Größe des Schaltkreises, und es lässt sich zeigen, daß es nicht möglich ist, eine äquivalente quantorenfreie Formel anzugeben, deren Länge nicht exponentiell in der Größe des Schaltkreises ist.

Dennoch läßt sich mit Hilfe der Einführung neuer Variablen das Problem Sat(Sk mit Par in $\mathcal{M}$) auf Sat(Qf Fml mit Par in $\mathcal{M}$) zurückführen:

Satz 4.2.9 *Das Problem* Sat(Qf Fml mit Par in $\mathcal{M}$) *ist* $\mathrm{NP}_{\mathcal{M}}$*-vollständig bezüglich* $\mathrm{P}_{\mathcal{M}}$*-Reduktion und* $\mathbb{NP}_{\mathcal{M}}$*-vollständig bezüglich* $\mathbb{P}_{\mathcal{M}}$*-Reduktion für alle Strukturen* $\mathcal{M}$*.*

Beweis: Das Problem Sat(Qf Fml mit Par in $\mathcal{M}$) liegt in $\mathrm{NP}_{\mathcal{M}}$, denn: Es läßt sich leicht eine deterministische Turingmaschine über $\mathcal{M}$ konstruieren, die bei Eingabe einer quantorenfreien Formel $\varphi(\bar{x})$ in binärer Kodierung und eines Elementes $\bar{a} \in M^{|\bar{x}|}$ in polynomieller Zeit überprüft, ob $\mathcal{M}$ Modell von $\varphi(\bar{a})$ ist: sie „rechnet $\varphi(\bar{a})$ aus“, wobei sie die entsprechenden Elemente für die in φ vorkommenden freien Variablen einsetzt, Funktionen einfach ausrechnet, für eine Relation R die entsprechende charakteristische Funktion χ_R einsetzt und auswertet und Junktoren mittels der Selektorfunktion simuliert. Eine Sat(Qf Fml mit Par in $\mathcal{M}$) akzeptierende nichtdeterministische Turingmaschine rät bei Eingabe von $\ulcorner\varphi(\bar{x}, \bar{y})\urcorner$ und $\bar{a} \in M^{|\bar{x}|}$ dann ein Element $\bar{b} \in M^{|\bar{y}|}$, überprüft, ob $\varphi(\bar{a}, \bar{b})$ in $\mathcal{M}$ gilt und gibt 1 aus, falls dies der Fall ist, 0 sonst.

Um zu zeigen, daß das Problem $\mathrm{NP}_{\mathcal{M}}$-vollständig bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion ist, zeigen wir:

$$\text{Sat(Sk mit Par in } \mathcal{M}) \leq^{\mathrm{P}_{\mathcal{M}}} \text{Sat(Qf Fml mit Par in } \mathcal{M})$$

Hierzu konstruieren wir eine deterministische Turingmaschine $\mathbb{A}$ im Sinne von $\mathcal{M}$, die eine Instanz von Sat(Sk mit Par in $\mathcal{M}$) in eine Instanz von Sat(Qf Fml mit Par in $\mathcal{M}$) umwandelt. Wie wir in der Vorüberlegung bereits gesehen haben, können wir bei Eingabe einer Kodierung eines Schaltkreises nicht einfach die durch diesen Schaltkreis repräsentierte quantorenfreie Formel berechnen, da diese zu groß sein könnte. Da wir am Ende eine Instanz von Sat(Qf Fml mit Par in $\mathcal{M}$) erhalten wollen, können wir aber neue Variablen einführen, die später existentiell quantifiziert werden.

Bei Eingabe von $\ulcorner C(\bar{x},\bar{y})\urcorner$ und $\bar{a}\in M^{|\bar{x}|}$, wobei $t(C)=l$ nehmen wir der Darstellung halber an, daß die Knoten von C durchnummeriert sind und führen für jeden Knoten i von C eine neue Variable z_i ein. Dann bilden wir für jeden Knoten i, beginnend bei den Eingangsknoten, folgendermaßen eine zugehörige Formel $\varphi_i(\bar{x},\bar{y},z_1,\ldots,z_l)$:

- Falls der Knoten i ein mit einer Variablen x_j gelabelter Eingangsknoten ist, ist $\varphi_i\equiv z_i=x_j$
- Falls der Knoten i ein mit einem k-stelligen Funktionssymbol f gelabelter Knoten ist und Pfeile von den Knoten $j_1,\ldots,j_k$ empfängt, ist $\varphi_i\equiv z_i=f(z_1,\ldots,z_k)$
- Falls der Knoten i ein mit χ_R für ein k-stelligen Relationssymbol R gelabelter Knoten ist und Pfeile von den Knoten $j_1,\ldots,j_k$ empfängt, ist $\varphi_i\equiv(Rz_1\ldots z_k\wedge z_i=1)\vee(Rz_1\ldots z_k\wedge z_i=0)$
- Falls der Knoten i der Ausgangsknoten ist, führen wir zusätzlich noch die Formel $\varphi_{out}\equiv z_i=1$ ein.

Die Formeln φ_i stellen sicher, daß die neuen Variablen denselben Wert wie der oberhalb des entsprechenden Knotens liegende Subschaltkreis haben. Durch φ_{out} wird gewährleistet, daß der Schaltkreis 1 ausgibt. Nun setzen wir

$$\varphi_C(\bar{x},\bar{y},\bar{z})\equiv\bigwedge_{i\in[l]}\varphi_i(\bar{x},\bar{y},\bar{z})\wedge\varphi_{out}(\bar{z})$$

Nach Konstruktion von φ_C gilt dann für alle $\bar{a}\in M^{|\bar{x}|},\bar{b}\in M^{|\bar{y}|}$:

$$C(\bar{a},\bar{b})=1\Leftrightarrow\mathcal{M}\models\exists\bar{z}\varphi_C(\bar{a},\bar{b},\bar{z})$$

Somit gilt für alle $\bar{a}\in M^{|\bar{x}|}$:

$$\text{es gibt ein }\bar{b}\in M^{|\bar{y}|}\text{ mit }C(\bar{a},\bar{b})=1\text{ genau dann, wenn }\mathcal{M}\models\exists\bar{y}\exists\bar{z}\varphi_C(\bar{a},\bar{y},\bar{z}).$$

Da in der Konstruktion für jeden Knoten höchstens konstant viele Schritte benötigt werden, ist $\ulcorner C\urcorner\mapsto\ulcorner\varphi_C\urcorner$ in polynomieller Zeit berechenbar. Die quantorenfreie Formel φ_C (oder vielmehr deren Kodierung $\ulcorner\varphi_C\urcorner$) zusammen mit den Parametern $\bar{a}$ ist nun eine Instanz von Sat(Qf Fml mit Par in $\mathcal{M}$). Somit gilt Sat(Sk mit Par in $\mathcal{M}$) $\leq^{\mathrm{P}_{\mathcal{M}}}$ Sat(Qf Fml mit Par in $\mathcal{M}$).

Die für eine $\mathbb{P}_{\mathcal{M}}$-Reduktion von Sat(Sk mit Par in $\mathcal{M}$) auf Sat(Qf Fml mit Par in $\mathcal{M}$) benötigte Schaltkreisfolge $(\widetilde{C}_n)_{n\in\mathbb{N}}$ konstruieren wir, indem wir die eben konstruierte Turingmaschine $\mathbb{A}$ leicht modifizieren und dann die Schaltkreise betrachten, die für die modifizierte Maschine $\mathbb{A}'$ die Übergangsfunktion berechnen. Dies ist notwendig, da die Ausgaben von $\mathbb{A}$ unterschiedliche Länge haben können.

Sei also $\mathbb{A}$ die oben konstruierte deterministische Turingmaschine und p das Polynom, welches die Laufzeit von $\mathbb{A}$ beschränkt. Die Maschine $\mathbb{A}'$ simuliert bei Eingabe von $\ulcorner C(\bar{x},\bar{y})\urcorner$ und $\bar{a}\in M^{|\bar{x}|}$ die Maschine $\mathbb{A}$. Nun muß die Ausgabe noch auf eine einheitliche Länge gebracht werden. Dazu schreibt $\mathbb{A}'$ noch sooft „$\wedge x_1=x_1$“ an den Schluß der Formel,

bis die Kodierung der so verlängerten Formel die Länge $p(|\ulcorner C(\bar{x},\bar{y})\urcorner| + |\bar{x}|) \cdot |\ulcorner \wedge x_1 = x_1 \urcorner|$ hat.

Sei nun für $n \in \mathbb{N}$ der Schaltkreis C'_n derjenige Schaltkreis, welcher die Übergangsfunktion der Maschine $\mathbb{A}'$ für Konfigurationen der Größe n beschreibt. Sei außerdem p' das Polynom, welches die Laufzeit von $\mathbb{A}'$ beschränkt. Für den Schaltkreis $\widetilde{C}_k$ schalten wir (analog zum Beweis von Satz 2.1.13 über die Darstellung von $\mathrm{P}_{\mathcal{M}}$ mittels einer in polynomieller Zeit konstruierbaren Schaltkreisfolge) $p'(k)$ Kopien von $C'_{p'(k)}$ hintereinander und sorgen abschließend mit Selektorknoten dafür, daß nur die ersten $p(k) \cdot |\ulcorner \wedge x_1 = x_1 \urcorner|$ Elemente der Ausgabe von $\mathbb{A}'$ auf Ausgangsknoten abgebildet werden. □

Mit Hilfe des Problems Sat(Qf Fml mit Par in $\mathcal{M}$) können wir nun eine Eigenschaft von Strukturen, in denen $\mathrm{NP}_{\mathcal{M}} = \mathrm{BNP}_{\mathcal{M}}$ gilt beweisen. Dies gilt dann speziell auch für Strukturen mit $\mathrm{P}_{\mathcal{M}} = \mathrm{NP}_{\mathcal{M}}$. Vorab eine kurze Erinnerung an einen Begriff aus der Modelltheorie.

Definition 4.2.10 (Quantorenelimination) Sei $\mathcal{M}$ eine Struktur. $\mathcal{M}$ *erlaubt Quantorenelimination* (genauer: die Theorie von $\mathcal{M}$ erlaubt Quantorenelimination), falls es für alle $n \in \mathbb{N}$ zu jeder Formel $\varphi(x_1,\ldots,x_n)$ eine quantorenfreie Formel $\psi(x_1,\ldots,x_n)$ gibt, so daß gilt:

$$\mathcal{M} \models \varphi(x_1,\ldots,x_n) \leftrightarrow \psi(x_1,\ldots,x_n)$$

Falls in einer Struktur $\mathcal{M}$ die Gleichheit $\mathrm{NP}_{\mathcal{M}} = \mathrm{BNP}_{\mathcal{M}}$ gilt, so liegt das Problem Sat(Qf Fml mit Par in $\mathcal{M}$), also die Frage, ob eine existentielle Formel in $\mathcal{M}$ gilt, in $\mathrm{BNP}_{\mathcal{M}}$.

Lemma 4.2.11 ($\mathrm{NP}_{\mathcal{M}} = \mathrm{BNP}_{\mathcal{M}} \Rightarrow$ QE$\mathcal{M}$) *Sei $\mathcal{M}$ eine Struktur mit $\mathrm{NP}_{\mathcal{M}} = \mathrm{BNP}_{\mathcal{M}}$ oder $\mathbb{NP}_{\mathcal{M}} = \mathbb{BNP}_{\mathcal{M}}$. Dann erlaubt $\mathcal{M}$ Quantorenelimination.*

Beweis: *i)* Wir zeigen die Behauptung zunächst für den Fall $\mathrm{NP}_{\mathcal{M}} = \mathrm{BNP}_{\mathcal{M}}$. Sei dazu $\mathcal{M}$ eine Struktur mit $\mathrm{NP}_{\mathcal{M}} = \mathrm{BNP}_{\mathcal{M}}$. Nach Satz 4.2.9 liegt das Problem der Erfüllbarkeit existentieller Formeln mit Parametern in $\mathcal{M}$ in den Klasse $\mathrm{NP}_{\mathcal{M}}$ und somit auch in $\mathrm{BNP}_{\mathcal{M}}$. Somit gilt also für jede existentielle Formel $\exists\bar{y}\varphi(\bar{a},\bar{y})$ mit Parametern $\bar{a}$:

$$\mathcal{M} \models \exists\bar{y}\varphi(\bar{a},\bar{y}) \Leftrightarrow (\ulcorner\varphi(\bar{x},\bar{y})\urcorner, \bar{a}) \in \text{Sat(Qf Fml mit Par in } \mathcal{M})$$

Es gibt also nach Satz 2.2.10 Polynome p und q und eine deterministische Turingmaschine $\mathbb{A}$ im Sinne von $\mathcal{M}$, so daß gilt:

$$\mathcal{M} \models \exists\bar{y}\varphi(\bar{a},\bar{y}) \quad \Leftrightarrow \quad \begin{array}{l}\text{es gibt ein } \bar{\varepsilon} \in \{0,1\}^{|\ulcorner\varphi\urcorner|+|\bar{a}|} \text{ so daß gilt:}\\ \mathbb{A} \text{ akzeptiert } (\ulcorner\varphi\urcorner, \bar{a}, \bar{\varepsilon}) \text{ in Zeit } q(|\ulcorner\varphi\urcorner| + |\bar{a}| + |\bar{\varepsilon}|)\end{array}$$

Nach Satz 2.1.13 kann die Maschine $\mathbb{A}$ als eine gesetzte Schaltkreisfolge $(C_n)_{n\in\mathbb{N}_{>0}}$ dargestellt werden, und es gilt:

$$\mathcal{M} \models \exists\bar{y}\varphi(\bar{a},\bar{y}) \Leftrightarrow \text{ es gibt ein } \bar{\varepsilon} \in \{0,1\}^{|\ulcorner\varphi\urcorner|+|\bar{a}|} \text{ mit: } C_{|\ulcorner\varphi\urcorner|+|\bar{a}|+|\bar{\varepsilon}|}(\ulcorner\varphi\urcorner, \bar{a}, \bar{\varepsilon}) = 1$$

Der Schaltkreis $C_{|\ulcorner\varphi\urcorner|+|\bar{a}|+|\bar{\varepsilon}|}$ repräsentiert eine quantorenfreie Formel ψ_C. Diese kann sehr lang sein, was uns jedoch in diesem Kontext nicht zu beunruhigen braucht. Somit erhalten wir

$$\mathcal{M} \models \exists\bar{y}\varphi(\bar{a},\bar{y}) \Leftrightarrow \text{ es gibt ein } \bar{\varepsilon} \in \{0,1\}^{|\ulcorner\varphi\urcorner|+|\bar{a}|} \text{ mit: } \mathcal{M} \models \psi_C(\ulcorner\varphi\urcorner,\bar{a},\bar{\varepsilon})$$

Da die Länge von $\bar{\varepsilon}$ fest ist, gibt es nur endlich viele verschiedene solcher $\bar{\varepsilon}$, und es gilt

$$\mathcal{M} \models \exists\bar{y}\varphi(\bar{a},\bar{y}) \Leftrightarrow \mathcal{M} \models \bigvee_{\bar{\varepsilon}\in\{0,1\}^{|\ulcorner\varphi\urcorner|+|\bar{a}|}} \psi_C(\ulcorner\varphi\urcorner,\bar{a},\bar{\varepsilon})$$

Diese Disjunktion ist also eine zu $\exists\bar{y}\varphi(\bar{a},\bar{y})$ äquivalente quantorenfreie Formel.

Da jede Formel zu einer Formel in pränexer Normalform äquivalent ist, und man „$\forall$“ durch „$\neg\exists\neg$“ ersetzen und somit mit der oben beschriebenen Methode nacheinander „von innen heraus“ alle Quantoren eliminieren kann, gibt es zu jeder Formel eine äquivalente quantorenfreie Formel, und die Struktur $\mathcal{M}$ erlaubt Quantorenelimination.

ii) Wenn $\mathcal{M}$ eine Struktur mit $\mathbb{NP}_\mathcal{M} = \mathbb{BNP}_\mathcal{M}$ ist, konstruieren wir die zu $\exists\bar{y}\varphi(\bar{a},\bar{y})$ äquivalente quantorenfreie Formel analog. Der einzige Unterschied ist, daß wir die Schaltkreisfolge $(C_n)_{n\in\mathbb{N}_{>0}}$ direkt aus den Definitionen von $\mathbb{NP}_\mathcal{M}$ und $\mathbb{P}_\mathcal{M}$ gewinnen. □

Bemerkung 4.2.12 Es gilt sogar: Wenn für ein $i \in \mathbb{N}_{>0}$ die Gleichheit $\Sigma_i\mathrm{P}_\mathcal{M} = \mathrm{B}\Sigma_i\mathrm{P}_\mathcal{M}$ gilt, so erlaubt $\mathcal{M}$ Quantorenelimination. Denn dann ist $\mathrm{NP}_\mathcal{M} \subseteq \Sigma_i\mathrm{P}_\mathcal{M}$ und somit $\mathrm{NP}_\mathcal{M} \subseteq \mathrm{B}\Sigma_i\mathrm{P}_\mathcal{M}$. Damit erhält man für Sat(Qf Fml mit Par in $\mathcal{M}$) eine Bedingung der Form: Für alle $\bar{x} \in M^*$ gilt

$$\bar{x} \in \text{Sat(Qf Fml mit Par in } \mathcal{M}) \Leftrightarrow \exists\bar{\varepsilon_1}\forall\bar{\varepsilon_2}\ldots Q_i\bar{\varepsilon_i} \text{ mit } C_{|\bar{\varepsilon_1}|+\ldots+|\bar{\varepsilon_i}|+|\bar{x}|}(\bar{x},\bar{\varepsilon_1},\ldots,\bar{\varepsilon_i}) = 1.$$

Mit Bemerkung 1.1.4 gibt es dann eine Familie von Formeln $(\psi_n)_{n\in\mathbb{N}_{>0}}$, so daß die rechte Seite genau dann der Fall ist, wenn gilt:

$$\exists\bar{\varepsilon_1}\forall\bar{\varepsilon_2}\ldots Q_i\bar{\varepsilon_i} \text{ mit } \mathcal{M} \models \psi_{C_{|\bar{\varepsilon_1}|+\ldots+|\bar{\varepsilon_i}|+|\bar{x}|}}(\bar{x},\bar{\varepsilon_1},\ldots,\bar{\varepsilon_i}).$$

Da die $\bar{\varepsilon_j}$ alle aus $\{0,1\}^*$ sind und eine feste Länge haben, kann man dies umschreiben als

$$\mathcal{M} \models \bigvee_{\bar{\varepsilon_1}} \bigwedge_{\bar{\varepsilon_2}} \ldots J_{\bar{\varepsilon_i}}\psi_{C_{|\bar{\varepsilon_1}|+\ldots+|\bar{\varepsilon_i}|+|\bar{x}|}}(\bar{x},\bar{\varepsilon}).$$

Wie oben lässt sich nun aus der Formel $\bigvee_{\bar{\varepsilon_1}} \bigwedge_{\bar{\varepsilon_2}} \ldots J_{\bar{\varepsilon_i}}\psi_{C_{|\bar{\varepsilon_1}|+\ldots+|\bar{\varepsilon_i}|+|\bar{x}|}}(\bar{x},\bar{\varepsilon})$ die die gesuchte quantorenfreie Formel bilden.

Die Frage, ob die polynomielle Hierarchie in Sinne einer Struktur $\mathcal{M}$ kollabiert, ist somit also nur für Strukturen, die Quantorenelimination erlauben, interessant. In Strukturen, die keine Quantorenelimination erlauben, gilt $\mathrm{P}_\mathcal{M} \neq \mathrm{NP}_\mathcal{M}$. Allerdings reicht die Tatsache, daß eine Struktur Quantorenelimination erlaubt, noch nicht aus um folgern zu können, daß in dieser Struktur $\mathrm{P}_\mathcal{M} = \mathrm{NP}_\mathcal{M}$ gilt. Es läßt sich zum Beispiel zeigen, daß Boolesche Algebren ohne Atome Quantorenelimination erlauben, und daß in ihnen $\mathrm{P}_\mathcal{M} \neq \mathrm{NP}_\mathcal{M}$ gilt (für die Idee des Beweises siehe zum Beispiel [22]). Um die Erfüllbarkeit existentieller Formeln (also das Problem Sat(Qf Fml mit Par in $\mathcal{M}$)) schnell lösen zu können, ist es notwendig, daß wir uns die zu einer existentiellen Formel äquivalente quantorenfreie Formel in einer kleinen Darstellung schnell beschaffen können. Dies führt auf den Begriff der effektiven Quantorenelimination.

Definition 4.2.13 (Effektive Quantorenelimination) Eine Struktur $\mathcal{M}$ *erlaubt effektive Quantorenelimination*, falls es eine deterministische Turingmaschine im Sinne von $\mathcal{M}$ gibt, welche bei Eingabe einer existentiellen Formel $\exists\bar{y}\varphi(\bar{x},\bar{y})$ in binärer Kodierung in polynomieller Zeit die Kodierung eines Schaltkreises $C_\varphi(\bar{x})$ bildet, so daß für alle $\bar{a} \in M^{|\bar{x}|}$ gilt:

$$\mathcal{M} \models \exists\bar{y}\varphi(\bar{a},\bar{y}) \Leftrightarrow C_\varphi(\bar{a}) = 1$$

Hiermit können wir nun Strukturen, in denen $\mathrm{P}_\mathcal{M} = \mathrm{NP}_\mathcal{M}$ gilt, charakterisieren:

Satz 4.2.14 ($\mathrm{P}_\mathcal{M} = \mathrm{NP}_\mathcal{M}$ bedeutet EQE $\mathcal{M}$) *Sei $\mathcal{M}$ eine Struktur. Dann gilt:*

$$\mathrm{P}_\mathcal{M} = \mathrm{NP}_\mathcal{M} \Leftrightarrow \mathcal{M} \textit{ erlaubt effektive Quantorenelimination.}$$

Beweis: „→“: Gelte $\mathrm{P}_\mathcal{M} = \mathrm{NP}_\mathcal{M}$. Dann gilt auch $\textsc{Sat}(\textsc{Qf Fml mit Par in } \mathcal{M}) \in \mathrm{P}_\mathcal{M}$. Mit Satz 2.1.13 gibt es dann eine gesetzte Familie $(C_n)_{n\in\mathbb{N}_{>0}}$ von Schaltkreisen, welche $\textsc{Sat}(\textsc{Qf Fml mit Par in } \mathcal{M})$ akzeptiert, so daß die Funktion $1^n \mapsto C_n$ in polynomieller Zeit berechenbar ist. Die Maschine $\mathbb{A}$, welche aus parameterfreien existentiellen Formeln die äquivalenten Schaltkreise bildet, berechnet bei Eingabe von $\ulcorner\exists\bar{y}\varphi(\bar{x},\bar{y})\urcorner$ zunächst $|\bar{x}|$. Da $1^{|\bar{x}|} \in M^{|\bar{x}|}$ gilt, ist $(\ulcorner\exists\bar{y}\varphi(\bar{x},\bar{y})\urcorner, 1^{|\bar{x}|})$ eine Instanz von $\textsc{Sat}(\textsc{Qf Fml mit Par in } \mathcal{M})$. Die Maschine $\mathbb{A}$ berechnet nun in polynomieller Zeit den Schaltkreis $C_{|\ulcorner\varphi(\bar{x},\bar{y})\urcorner|+|\bar{x}|}(\bar{z},\bar{x})$, wobei $|\bar{z}| = |\ulcorner\varphi(\bar{x},\bar{y})\urcorner|$. Für den Schaltkreis $C_{|\ulcorner\varphi(\bar{x},\bar{y})\urcorner|+|\bar{x}|}(\ulcorner\varphi(\bar{x},\bar{y})\urcorner,\bar{x})$ gilt dann für alle $\bar{a} \in M^{|\bar{x}|}$:

$$\mathcal{M} \models \exists\bar{y}\varphi(\bar{a},\bar{y}) \Leftrightarrow C_{|\ulcorner\varphi(\bar{x},\bar{y})\urcorner|+|\bar{x}|}(\ulcorner\varphi(\bar{x},\bar{y})\urcorner,\bar{a}) = 1$$

Die Maschine gibt also $\ulcorner C_{|\ulcorner\varphi(\bar{x},\bar{y})\urcorner|+|\bar{x}|}(\ulcorner\varphi(\bar{x},\bar{y})\urcorner,\bar{x})\urcorner$ aus.
„←“: Wenn andererseits $\mathcal{M}$ effektive Quantorenelimination erlaubt, so lässt sich folgendermaßen eine deterministische Turingmaschine $\mathbb{A}'$ im Sinne von $\mathcal{M}$ konstruieren, welche das Problem $\textsc{Sat}(\textsc{Qf Fml mit Par in } \mathcal{M})$ in polynomieller Zeit entscheidet. Da $\textsc{Sat}(\textsc{Qf Fml mit Par in } \mathcal{M})$ nach Satz 4.2.9 $\mathrm{NP}_\mathcal{M}$-vollständig bezüglich $\mathrm{P}_\mathcal{M}$-Reduktion ist, gilt damit auch $\mathrm{P}_\mathcal{M} = \mathrm{NP}_\mathcal{M}$. Bei Eingabe von $(\ulcorner\varphi(\bar{x},\bar{y})\urcorner,\bar{a})$ simuliert $\mathbb{A}'$ nun die Maschine, welche in polynomieller Zeit den zu $\exists\bar{y}\varphi(\bar{x},\bar{y})$ äquivalenten Schaltkreis $C(\bar{x})$ bildet. Nun wertet sie $C(\bar{a})$ aus (was nach Satz 2.1.6 in polynomieller Zeit möglich ist) und gibt 1 aus, falls $C(\bar{a}) = 1$, andernfalls 0. □

In [24] wird eine Struktur beschrieben, welche effektive Quantorenelimination erlaubt, und in welcher somit $\mathrm{P}_\mathcal{M} = \mathrm{NP}_\mathcal{M}$ gilt.

Betrachten wir nun ein weiteres vollständiges Problem, das Problem der Erfüllbarkeit von rudimentären Formeln über einer Struktur $\mathcal{M}$.

Definition 4.2.15 (Rudimentäre Formeln) Eine Formel $\varphi(\bar{x})$ der Sprache von $\mathcal{M}$ heißt *rudimentäre Formel über $\mathcal{M}$*, falls φ eine endliche Konjunktion von Formeln $\psi_l(\bar{x})$ der Form

(i) $x_i = x_j$ mit $i,j \in [|\bar{x}|]$

(ii) $f(x_{i_1},\ldots,x_{i_n}) = x_j$ für eine n-stellige Funktion f aus $\mathcal{M}$ und $i_1,\ldots,i_n,j \in [|\bar{x}|]$

(iii) $Rx_{i_1} \dots x_{i_n} \vee x_j = 0$ oder $\neg Rx_{i_1} \dots x_{i_n} \vee x_j = 1$ für eine n-stellige Relation R aus $\mathcal{M}$ und $i_1, \dots, i_n, j \in [|\bar{x}|]$, wobei R die Gleichheit sein kann

(iv) $x_i = \varepsilon \vee x_j = \varepsilon' \vee x_k = \varepsilon''$ mit $\varepsilon, \varepsilon', \varepsilon'' \in \{0, 1\}$ und $i, j, k \in [|\bar{x}|]$

ist.

Hieraus ergibt sich natürlicherweise das Problem der Erfüllbarkeit rudimentärer Formeln mit Parametern aus M:

SAT(RUD FML MIT PAR IN $\mathcal{M}$)
Eingabe: Eine rudimentäre Formel $\varphi(\bar{x}, \bar{y})$ in binärer Kodierung, $\bar{a} \in M^{\lvert\bar{x}\rvert}$
Frage: Gibt es ein $\bar{b} \in M^{\lvert\bar{y}\rvert}$ mit $\mathcal{M} \models \varphi(\bar{a}, \bar{b})$?

Das folgende Lemma über die $\mathrm{NP}_{\mathcal{M}}$-Vollständigkeit bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion und die $\mathbb{NP}_{\mathcal{M}}$-Vollständigkeit bezüglich $\mathbb{P}_{\mathcal{M}}$-Reduktion von SAT(RUD FML MIT PAR IN $\mathcal{M}$) ist eher von technischem Nutzen.

Lemma 4.2.16 (Vollständigkeit von satrud) *Sei $\mathcal{M}$ eine Struktur. Dann ist das Problem* SAT(RUD FML MIT PAR IN $\mathcal{M}$) *bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\mathrm{NP}_{\mathcal{M}}$-vollständig und $\mathbb{NP}_{\mathcal{M}}$-vollständig bezüglich $\mathbb{P}_{\mathcal{M}}$-Reduktion.*

Beweis: Zunächst ist SAT(RUD FML MIT PAR IN $\mathcal{M}$) in $\mathrm{NP}_{\mathcal{M}}$, denn eine nichtdeterministische Turingmaschine im Sinne von $\mathcal{M}$, welche bei Eingabe von $(\ulcorner\varphi(\bar{x}, \bar{y})\urcorner, \bar{a})$ ein $\bar{b} \in M^{\lvert\bar{y}\rvert}$ rät, dann $\varphi(\bar{a}, \bar{b})$ in polynomieller Zeit auswertet und 1 ausgibt, falls $\mathcal{M} \models \varphi(\bar{a}, \bar{b})$, andernfalls 0, akzeptiert das Problem SAT(RUD FML MIT PAR IN $\mathcal{M}$).

Für die $\mathrm{NP}_{\mathcal{M}}$-Vollständigkeit bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion zeigen wir, daß

$$\text{SAT(QF FML MIT PAR IN } \mathcal{M}) \leq^{\mathrm{P}_{\mathcal{M}}} \text{SAT(RUD FML MIT PAR IN } \mathcal{M})$$

in allen Strukturen $\mathcal{M}$ gilt. Da in Satz 4.2.9 die $\mathrm{NP}_{\mathcal{M}}$-Vollständigkeit bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion von SAT(QF FML MIT PAR IN $\mathcal{M}$) gezeigt wurde, gilt somit auch die $\mathrm{NP}_{\mathcal{M}}$-Vollständigkeit bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion von SAT(RUD FML MIT PAR IN $\mathcal{M}$).

Betrachten wir also eine Instanz $(\ulcorner\varphi(\bar{x}, \bar{y})\urcorner, \bar{a})$ von SAT(QF FML MIT PAR IN $\mathcal{M}$). Um diese in eine Instanz von SAT(RUD FML MIT PAR IN $\mathcal{M}$) umzuwandeln, lösen wir zunächst die Terme der atomaren Subformeln von φ auf, indem wir Hilfsvariablen einführen. Dann legen wir mit Hilfe von zusätzlichen Variablen induktiv die Wahrheitswerte der Subformeln von φ fest.

Einen verschachtelten Term t der Form $f(t_1(\bar{x}, \bar{y}), \dots, t_n(\bar{x}, \bar{y}))$ lösen wir auf, indem wir für jeden der Terme t_i und für t eine neue Variable z_{t_i} beziehungsweise z_t einfügen und die Konjunktion

$$\bigwedge_{i=1}^{n} t_i(\bar{x}, \bar{y}) = z_{t_i} \wedge f(z_{t_1}, \dots, z_{t_n}) = z_t$$

bilden. Dies liefert Konjunktionsglieder der Form (i) und (ii) der Definition von rudimentären Formeln.

Um nun induktiv die Wahrheitswerte der Subformeln von φ festzulegen, beginnen wir mit den atomaren Subformeln. Eine atomare Subformel ψ hat für eine n-stellige Relation

R aus $\mathcal{M}$, welche die Gleichheit sein kann, und Terme $t_1, \ldots, t_n$ die Form $\psi = Rt_1 \ldots t_n$. Wir führen eine neue Variable u_ψ für den Wahrheitswert von ψ ein und bilden mit Hilfe der im ersten Schritt für die Terme $t_1, \ldots, t_n$ hinzugefügten Variablen $z_{t_1}, \ldots, z_{t_n}$ die Konjunktion

$$(Rz_{t_1} \ldots z_{t_n} \vee u_\psi = 0) \wedge (\neg Rz_{t_1} \ldots z_{t_n} \vee u_\psi = 1).$$

Diese liefert Konjunktionsglieder der Form (iii) der Definition von rudimentären Formeln.

Um die Wahrheitswerte der nichtatomaren Subformeln von φ festzulegen, gehen wir induktiv vor: wenn wir die Wahrheitswerte von zwei Subformeln ψ_1 und ψ_2 schon festgelegt haben, und die Formel $\psi = \psi_1 \vee \psi_2$ eine Subformel von φ ist, fügen wir die neue Variable u_ψ hinzu und bilden die Konjunktion

$$(u_\psi = 1 \vee u_{\psi_1} = 0 \vee u_{\psi_1} = 0) \wedge (u_\psi = 1 \vee u_{\psi_2} = 0 \vee u_{\psi_2} = 0) \wedge (u_\psi = 0 \vee u_{\psi_1} = 1 \vee u_{\psi_2} = 1).$$

Ist andererseits die Formel $\psi = \psi_1 \wedge \psi_2$ eine Subformel von φ, so bilden wir die Konjunktion

$$(u_\psi = 0 \vee u_{\psi_1} = 1 \vee u_{\psi_1} = 1) \wedge (u_\psi = 0 \vee u_{\psi_2} = 1 \vee u_{\psi_2} = 1) \wedge (u_\psi = 1 \vee u_{\psi_1} = 0 \vee u_{\psi_2} = 0).$$

Ist schließlich die Formel $\psi = \neg\psi_1$ eine Subformel von φ, so bilden wir die Konjunktion

$$(u_\psi = 0 \vee u_{\psi_1} = 0 \vee u_{\psi_1} = 0) \wedge (u_\psi = 1 \vee u_{\psi_1} = 1 \vee u_{\psi_1} = 1).$$

Dies liefert Konjunktionsglieder der Form (iv) der Definition von rudimentären Formeln. Da die Formel erfüllt werden soll, fügen wir für die Wahrheitswertvariable u_φ von φ noch das Konjunktionsglied $1 = u_\varphi$, also ein Konjunktionsglied der Form (iii) der Definition von rudimentären Formeln. Die Konjunktion der auf diese Weise konstruierten Formeln ist dann eine rudimentäre Formel φ', für die für alle $\bar{x}, \bar{y} \in M^*$ gilt:

$$\mathcal{M} \models \varphi(\bar{x}, \bar{y}) \Leftrightarrow \mathcal{M} \models \exists\bar{u}\exists\bar{z}\varphi'(\bar{x}, \bar{y}, \bar{u}, \bar{z}).$$

Somit gilt auch für alle $\bar{a} \in M^*$:

$$\mathcal{M} \models \exists\bar{y}\varphi(\bar{a}, \bar{y}) \Leftrightarrow \mathcal{M} \models \exists\bar{y}\exists\bar{u}\exists\bar{z}\varphi'(\bar{a}, \bar{y}, \bar{u}, \bar{z})$$

und die Frage, ob $(\ulcorner\varphi\urcorner, \bar{a})$ in SAT(QF FML MIT PAR IN $\mathcal{M}$) liegt, ist äquivalent zu der Frage, ob $(\ulcorner\varphi'\urcorner, \bar{a})$ in SAT(RUD FML MIT PAR IN $\mathcal{M}$) liegt. Da die durchgeführte Konstruktion der rudimentären Formel φ' in polynomieller Zeit bewerkstelligt werden kann, haben wir eine $\mathrm{P}_\mathcal{M}$-Reduktion von SAT(QF FML MIT PAR IN $\mathcal{M}$) auf das Problem SAT(RUD FML MIT PAR IN $\mathcal{M}$). □

Lässt man bei der Definition von rudimentären Formeln die Disjunktionen weg, so erhält man rudimentär-plus Formeln:

Definition 4.2.17 (Rudimentär-plus Formeln) Sei $\mathcal{M}$ eine Struktur. Eine Formel $\varphi(\bar{x}, \bar{y})$ der Sprache von $\mathcal{M}$ heißt *rudimentär-plus Formel über* $\mathcal{M}$, falls sie eine endliche Konjunktion von Formeln der Form

(i) $x_i = x_j$ mit $i, j \in [|\bar{x}|]$

(ii) $f(x_{i_1}, \ldots, x_{i_n}) = x_j$ für eine n-stellige Funktion f aus $\mathcal{M}$ und $i_1, \ldots, i_n, j \in [|\bar{x}|]$

(iii) $Rx_{i_1} \ldots x_{i_n}$ oder $\neg Rx_{i_1} \ldots x_{i_n}$ für eine n-stellige Relation R aus $\mathcal{M}$ und $i_1, \ldots, i_n \in [|\bar{x}|]$, wobei R die Gleichheit sein kann

ist.

Auch hier betrachten wir wieder das Problem der Erfüllbarkeit solcher Formeln mit Parametern:

SAT(RUD+ FML MIT PAR IN $\mathcal{M}$)
Eingabe: Eine rudimentär-plus Formel $\varphi(\bar{x}, \bar{y})$ in binärer Kodierung, $\bar{a} \in M^{|\bar{x}|}$
Frage: Gibt es ein $\bar{b} \in M^{|\bar{y}|}$ mit $\mathcal{M} \models \varphi(\bar{a}, \bar{b})$?

Mit Hilfe des Problems SAT(RUD+ FML MIT PAR IN $\mathcal{M}$) können wir nun ein Kriterium für $\mathrm{NP}_\mathcal{M} = \mathrm{BNP}_\mathcal{M}$ angeben. Die Frage, ob in einer Struktur $\mathrm{NP}_\mathcal{M} = \mathrm{BNP}_\mathcal{M}$ gilt, ist wichtig in Hinblick auf die Frage, ob sich diese Struktur bezüglich der $\mathrm{P}_\mathcal{M} = \mathrm{NP}_\mathcal{M}$-Frage genauso verhält, wie die Standardstruktur.

Lemma 4.2.18 ($\mathrm{NP}_\mathcal{M} = \mathrm{BNP}_\mathcal{M}$ gdw sat(rud+)$\in \mathrm{BNP}_\mathcal{M}$) *Für jede Struktur $\mathcal{M}$ gilt:*

$$\mathrm{NP}_\mathcal{M} = \mathrm{BNP}_\mathcal{M} \Leftrightarrow \text{SAT(RUD+ FML MIT PAR IN } \mathcal{M}) \in \mathrm{BNP}_\mathcal{M}.$$

Beweis: „$\rightarrow$“: Das Problem SAT(RUD+ FML MIT PAR IN $\mathcal{M}$) ist in $\mathrm{NP}_\mathcal{M}$, denn eine Maschine, die überprüft, ob eine Eingabe $\ulcorner\varphi\urcorner$ eine Kodierung einer rudimentär-plus Formel ist, dann erfüllende Elemente $\bar{b}$ rät und $\varphi(\bar{a}, \bar{b})$ auswertet, akzeptiert gerade das Problem SAT(RUD+ FML MIT PAR IN $\mathcal{M}$) in polynomieller Zeit.

„$\leftarrow$“: Wir zeigen, daß falls SAT(RUD+ FML MIT PAR IN $\mathcal{M}) \in \mathrm{BNP}_\mathcal{M}$ gilt, das Problem SAT(RUD FML MIT PAR IN $\mathcal{M}$) in $\mathrm{BNP}_\mathcal{M}$ liegt. Da SAT(RUD FML MIT PAR IN $\mathcal{M}$) nach Lemma 4.2.16 bezüglich $\mathrm{P}_\mathcal{M}$-Reduktion $\mathrm{NP}_\mathcal{M}$-vollständig ist, gilt damit $\mathrm{NP}_\mathcal{M} = \mathrm{BNP}_\mathcal{M}$.

Sei also $\mathcal{M}$ eine Struktur mit SAT(RUD+ FML MIT PAR IN $\mathcal{M}) \in \mathrm{BNP}_\mathcal{M}$, und sei $\mathbb{A}$ eine binär nichtdeterministische Turingmaschine über $\mathcal{M}$, die SAT(RUD+ FML MIT PAR IN $\mathcal{M}$) in polynomieller Zeit akzeptiert. Wir konstruieren eine binär nichtdeterministische Turingmaschine $\mathbb{A}'$ über $\mathcal{M}$, welche SAT(RUD FML MIT PAR IN $\mathcal{M}$) in polynomieller Zeit akzeptiert. Bei Eingabe von $(\ulcorner\varphi(\bar{x}, \bar{y})\urcorner, \bar{a})$, wobei φ eine rudimentäre Formel über $\mathcal{M}$ ist, rät $\mathbb{A}'$ zunächst eine 0-1-Folge der Länge $|\ulcorner\varphi(\bar{x}, \bar{y})\urcorner| + |\bar{a}|$. Diese Folge wird interpretiert als Angabe darüber, welches der jeweils höchstens drei Disjunkte aus jedem Konjunkt von φ erfüllt werden soll. Nun konstruiert $\mathbb{A}'$ eine rudimentär-plus Formel $\psi(\bar{x}, \bar{y})$ aus φ, indem sie jeweils abhängig von den entsprechenden Elementen der geratenen 0-1-Folge ein Disjunkt aus jedem Konjunkt von φ übernimmt. Als letztes simuliert $\mathbb{A}'$ die Maschine $\mathbb{A}$ angesetzt auf $(\ulcorner\psi(\bar{x}, \bar{y})\urcorner, \bar{a})$ und gibt das Ergebnis aus. □

4.3 Die Standardtypen

Unser nächstes Ziel ist es, Eigenschaften von Strukturen zusammenzustellen, welche sicherstellen, daß die diese Eigenschaften erfüllenden Strukturen sich in bezug auf einen Kollaps der polynomiellen Hierarchie ähnlich wie die Standardstruktur verhalten. Diese Eigenschaften werden wir in den Begriffen der Standardtypen zusammenfassen. Wir einigen uns zunächst auf eine Schreibweise:

Schreibweise 4.3.1 Für eine Struktur $\mathcal{M}$, eine Menge $N \subseteq M$ und eine Menge $\mathcal{C}$ von Problemen aus M^* sei

$$\mathcal{C} \restriction N := \{X \cap N^* \mid X \in \mathcal{C}\}.$$

Da unseren Turingmaschinen nur die Verwendung der Konstanten der entsprechenden Struktur erlaubt ist, erhalten wir, daß Turingmaschinen im Sinne einer Struktur $\mathcal{M}$ auch Turingmaschinen im Sinne von jeder Substruktur von $\mathcal{M}$ sind und damit:

Lemma 4.3.2 *Seien $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $\mathcal{M} \subseteq \mathcal{N}$. Dann gilt:*

$$\mathrm{P}_{\mathcal{N}} \restriction M = \mathrm{P}_{\mathcal{M}}$$

Beweis: „$\subseteq$“: Sei $X \in \mathrm{P}_{\mathcal{N}} \restriction M$. Dann ist $X \subseteq M^*$ und es gibt eine deterministische Turingmaschine $\mathbb{A}$ im Sinne von $\mathcal{N}$, die X in polynomieller Zeit enscheidet. Da $\mathcal{M}$ eine Substruktur von $\mathcal{N}$ ist, ist $\mathbb{A}$ auch eine deterministische Turingmaschine im Sinne von $\mathcal{M}$. Somit ist $X \in \mathrm{P}_{\mathcal{M}}$.

„$\supseteq$“: Sei andersherum $X \in \mathrm{P}_{\mathcal{M}}$ und sei $\mathbb{A}$ die deterministische Turingmaschine im Sinne von $\mathcal{M}$, die X in Zeit $p(n)$ entscheidet. Wir konstruieren eine deterministische Turingmaschine $\mathbb{A}'$ im Sinne von $\mathcal{N}$, indem wir $\mathbb{A}'$ bei Eingabe von $\bar{x} \in N^*$ zunächst den Wert $p(|\bar{x}|)$ berechnen und anschließend die ersten $p(|\bar{x}|)$ Schritte von $\mathbb{A}$ simulieren lassen. Hält $\mathbb{A}$ innerhalb dieser Zeit, so hält auch $\mathbb{A}'$ und gibt das Ergebnis der Berechnung von $\mathbb{A}$ aus. Hält $\mathbb{A}$ nicht innerhalb von $p(|\bar{x}|)$ Schritten, so hält $\mathbb{A}'$ und gibt 0 aus. Sei $Y := \{\bar{x} \in N^* \mid \mathbb{A}'(\bar{x}) = 1\}$. Nach Konstruktion gilt $Y \in \mathrm{P}_{\mathcal{N}}$ und $Y \cap M^* = X$. Somit gilt $X \in \mathrm{P}_{\mathcal{N}} \restriction M$. □

Nun können wir das etwas abstraktere folgende Lemma formulieren.

Lemma 4.3.3 *Seien $\mathcal{M}_1$ und $\mathcal{M}_2$ Strukturen mit $M_1 \subseteq M_2$ und sei $R \subseteq M_1$. Gelte $\mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}_2} \restriction M_1 = \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}_1}$ für ein $i \in \mathbb{N}$. Dann gilt für alle $j \in \mathbb{N}$ mit $j > i$:*

- $\mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}_2} \restriction M_1 = \mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}_1}$
- $\mathrm{R}\Pi_j\mathrm{P}_{\mathcal{M}_2} \restriction M_1 = \mathrm{R}\Pi_j\mathrm{P}_{\mathcal{M}_1}$

Beweis: Da nach Definition $\mathrm{R}\Pi_j\mathrm{P}_{\mathcal{M}_2} = co\mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}_2}$ gilt, reicht es, $\mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}_2} \restriction M_1 = \mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}_1}$ für alle $j > i$ zu zeigen. Sei also $j > i$ beliebig. Für ein Problem X aus $\mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}_2} \restriction M_1$ gibt es Polynome $p_1(n) \ldots p_j(n)$ und ein $Y \in \mathrm{P}_{\mathcal{M}_2}$, so daß für alle $\bar{x} \in M_2^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \bar{x} \in M_1^* \text{ und } \exists \bar{y_1} \in R^{p_1(|\bar{x}|)} \forall \bar{y_2} \in R^{p_2(|\bar{x}|)} \ldots Q_j \bar{y_j} \in R^{p_j(|\bar{x}|)} \text{ mit } \bar{x}\bar{y_1} \ldots \bar{y_j} \in Y.$$

Betrachten wir das Problem $Y' \subseteq M_2^*$, welches gegeben ist durch:

$$Y' = \{\bar{y} \in M_2^* \mid Q_{j-(i-1)}\overline{y_{j-(i-1)}}Q_{j-(i-2)}\overline{y_{j-(i-2)}} \ldots, Q_j\bar{y_j} \text{ mit } \bar{x}\overline{y_{j-(i-1)}} \ldots \bar{y_j} \in Y\}.$$

Dann gilt für alle $\bar{x} \in M_2^*$:

$$\bar{x} \in X \Leftrightarrow \exists \bar{y_1} \in R^{p_1(|\bar{x}|)} \forall \bar{y_2} \in R^{p_2(|\bar{x}|)} \ldots Q_{j-i}\bar{y_{j-i}} \in R^{p_j(|\bar{x}|)} \text{ mit } \bar{x}\bar{y_1} \ldots \bar{y_{j-i}} \in Y' \cap M_1^*.$$

Das Problem $Y' \cap M_1^*$ ist dabei für ungerades $(j-i)$ in $\mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}_2} \restriction M_1$ und für gerades $(j-i)$ in $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}_2} \restriction M_1$. Da nach Voraussetzung

$$\mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}_2} \restriction M_1 = \mathrm{R}\Sigma_i\mathrm{P}_{\mathcal{M}_1}$$

und somit auch

$$\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}_2} \restriction M_1 = \mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}_1}$$

gilt, ist $Y' \cap M_1^*$ für ungerades $(j-i)$ in der Klasse $\mathrm{R}\Sigma_i\mathrm{P}\mathcal{M}_1$ und für gerades $(j-i)$ in der Klasse $\mathrm{R}\Pi_i\mathrm{P}_{\mathcal{M}_1}$. Damit gibt es aber ein $Y'' \in \mathrm{P}_{\mathcal{M}_1}$ und Polynome $q_1(n), \ldots, q_j(n)$, so daß für alle $\bar{x} \in M_1^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \exists \bar{y_1} \in R^{q_1(|\bar{x}|)} \ldots Q_j\bar{y_j} \in R^{q_j(|\bar{x}|)} \text{ mit } \bar{x}\bar{y_1} \ldots \bar{y_j} \in Y''.$$

Das Problem X ist also in der Klasse $\mathrm{R}\Sigma_j\mathrm{P}_{\mathcal{M}_1}$. □

Als ein Korollar erhalten wir, daß sich ein Kollaps der booleschen polynomiellen Hierarchie auf die i-ten Ebene, insbesondere also auch die Gleichheit $\mathrm{P}_{\mathcal{M}} = \mathrm{BNP}_{\mathcal{M}}$, auf Substrukturen vererbt:

Korollar 4.3.4 *Seien $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $\mathcal{M} \subseteq \mathcal{N}$. Dann gilt für alle $i \in \mathbb{N}_0$:*

$$\mathrm{BPH}_{\mathcal{N}} = \mathrm{B}\Sigma_i\mathrm{P}_{\mathcal{N}} \Rightarrow \mathrm{BPH}_{\mathcal{M}} = \mathrm{B}\Sigma_i\mathrm{P}_{\mathcal{M}}$$

Beweis: Gelte also $\mathcal{M} \subseteq \mathcal{N}$ und $\mathrm{BPH}_{\mathcal{N}} = \mathrm{B}\Sigma_i\mathrm{P}_{\mathcal{N}}$ für ein $i \in \mathbb{N}$. Da $\mathcal{M}$ eine Substruktur von $\mathcal{N}$ ist, gilt mit Lemma 4.3.2 die Gleichheit $\mathrm{P}_{\mathcal{N}} \restriction M = \mathrm{P}_{\mathcal{M}}$ und somit auch:

$$\mathrm{B}\Sigma_0\mathrm{P}_{\mathcal{N}} \restriction M = \mathrm{B}\Sigma_0\mathrm{P}_{\mathcal{M}}.$$

Mit dem vorangegangen Lemma erhalten wir also $\mathrm{B}\Sigma_i\mathrm{P}_{\mathcal{M}} = \mathrm{B}\Sigma_i\mathrm{P}_{\mathcal{N}} \restriction M$ und $\mathrm{B}\Sigma_j\mathrm{P}_{\mathcal{M}} = \mathrm{B}\Sigma_j\mathrm{P}_{\mathcal{N}} \restriction M$ für alle $j > i$. Da nach Voraussetzung die boolesche polynomielle Hierarchie im Sinne von $\mathcal{N}$ auf die i-te Ebene kollabiert, gilt für alle $j > i$ die Gleichheit $\mathrm{B}\Sigma_j\mathrm{P}_{\mathcal{N}} \restriction M = \mathrm{B}\Sigma_i\mathrm{P}_{\mathcal{N}} \restriction M$ und somit auch

$$\mathrm{B}\Sigma_j\mathrm{P}_{\mathcal{M}} = \mathrm{B}\Sigma_i\mathrm{P}_{\mathcal{M}}.$$

Mit Korollar 4.1.12 kollabiert dann die boolesche polynomielle Hierarchie im Sinne von $\mathcal{M}$ auf die i-te Ebene. □

Insbesondere vererbt sich also auch die Ungleichheit $\mathrm{P}_{\mathcal{M}} \neq \mathrm{BNP}_{\mathcal{M}}$ auf Oberstrukturen.

Im allgemeinen ist dagegen für eine Struktur $\mathcal{N}$ und eine Substruktur $\mathcal{M}$ von $\mathcal{N}$ ein Problem aus einer Klasse $\Sigma_i\mathrm{P}_{\mathcal{M}}$ nicht automatisch in der Klasse $\Sigma_i\mathrm{P}_{\mathcal{N}}$, da es zum Beispiel für ein $\bar{x} \in M^*$ keinen Zeugen aus M^*, wohl aber einen Zeugen aus N^* für die Zugehörigkeit zu X geben kann. Somit ist im allgemeinen nicht klar, ob sich ein Kollaps der polynomiellen Hierarchie auf Substrukturen vererbt. Allerdings gilt:

Lemma 4.3.5 *Seien $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $\mathcal{M} \subseteq \mathcal{N}$ und sei $M \in \Sigma_i \mathrm{P}_{\mathcal{N}}$ für ein $i \in \mathbb{N}_{>0}$. Dann gilt:*

(i) $\mathrm{PH}_{\mathcal{N}} = \mathrm{P}_{\mathcal{N}} \Rightarrow \mathrm{PH}_{\mathcal{M}} = \mathrm{P}_{\mathcal{M}}$

(ii) für alle $R \subseteq M$ und alle $j \in \mathbb{N}$ mit $j > i$ gilt: $\mathrm{R}\Sigma_j \mathrm{P}_{\mathcal{N}} = \Sigma_j \mathrm{P}_{\mathcal{N}} \Rightarrow \mathrm{R}\Sigma_j \mathrm{P}_{\mathcal{M}} = \Sigma_j \mathrm{P}_{\mathcal{M}}$

Beweis: *Zu (i):* Gelte also $\mathrm{PH}_{\mathcal{N}} = \mathrm{P}_{\mathcal{N}}$. Wir zeigen, daß $\mathrm{NP}_{\mathcal{M}} \subseteq \mathrm{P}_{\mathcal{M}}$ gilt. Mit Korollar 4.1.13 gilt dann $\mathrm{PH}_{\mathcal{M}} = \mathrm{P}_{\mathcal{M}}$. Sei $X \in \mathrm{NP}_{\mathcal{M}}$. Dann gibt es ein Polynom $p(n)$ und ein Problem $Y \in \mathrm{P}_{\mathcal{M}}$ und für alle $\bar{x} \in M^*$ gilt:

$$\begin{aligned} \bar{x} \in X &\Leftrightarrow \exists \bar{y} \in M^{p(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in Y \\ &\Leftrightarrow \exists \bar{y} \in N^{p(|\bar{x}|)} \text{ mit } \bar{y} \in M^* \text{ und } \bar{x}\bar{y} \in Y \end{aligned}$$

Da $M \in \Sigma_i \mathrm{P}_{\mathcal{N}}$ gilt, gibt es andererseits ein $k \in \mathbb{N}$ und ein $Y' \in \mathrm{P}_{\mathcal{N}}$, so daß für alle $y \in N^*$ gilt:

$$y \in M \Leftrightarrow \exists \bar{z_1} \in N^k \ldots Q_i \bar{z_i} \in N^k \text{ mit } \bar{y}\bar{z} \in Y'.$$

Um die Zugehörigkeit von $\bar{x}$ zu X mit Hilfe eines Zeugen $\bar{y} \in N^{p(|\bar{x}|)}$ über $\mathcal{N}$ zu verifizieren, muß für jede Komponente von $\bar{y}$ und für jede Komponente von $\bar{x}$ geprüft werden, ob sie aus M ist. Die dafür benötigten $ik \cdot (p(|\bar{x}|) + |\bar{x}|)$ quantifizierten Variablen können wir derart umstellen, daß wir i abwechselnd existentiell und universell quantifizierte Blöcke mit jeweils $k \cdot (p(|\bar{x}|) + |\bar{x}|)$ Variablen erhalten. Die $p(|\bar{x}|)$ existentiell quantifizierten Variablen des Zeugen schreiben wir vor den existentiell quantifizierten ersten Block. Somit hat dieser nun die Länge $k \cdot (p(|\bar{x}|) + |\bar{x}|) + p(|\bar{x}|)$, die Längen der anderen Blöcke bleiben unverändert.

Das Nachprüfen der Zugehörigkeit zu M benötigt für die $|\bar{x}| + p(|\bar{x}|)$ zu überprüfenden Elemente jeweils polynomielle Zeit in $|\bar{x}|$, also auch insgesamt nur polynomielle Zeit in $|\bar{x}|$. Ebenso ist die Zugehörigkeit von $\bar{x}$ zusammen mit dem Zeugen zu Y in polynomieller Zeit entscheidbar. Somit gilt aber $X \in \Sigma_i \mathrm{P}_{\mathcal{N}}$ und, da auch die Zugehörigkeit von $\bar{x}$ zu M^* geprüft wird, sogar $X \in \Sigma_i \mathrm{P}_{\mathcal{N}} \restriction M$. Da nach Voraussetzung $\mathrm{PH}_{\mathcal{N}} = \mathrm{P}_{\mathcal{N}}$ und somit auch $\Sigma_i \mathrm{P}_{\mathcal{N}} = \mathrm{P}_{\mathcal{N}}$ gilt, gilt dann

$$X \in \mathrm{P}_{\mathcal{N}} \restriction M.$$

Da $\mathcal{M}$ eine Substruktur von $\mathcal{N}$ ist, ist aber $\mathrm{P}_{\mathcal{N}} \restriction M = \mathrm{P}_{\mathcal{M}}$ und somit auch $X \in \mathrm{P}_{\mathcal{M}}$. Da $X \in \Sigma_i \mathrm{P}_{\mathcal{M}}$ beliebig war, gilt damit $\Sigma_i \mathrm{P}_{\mathcal{M}} = \mathrm{P}_{\mathcal{M}}$ und mit Korollar 4.1.13 folgt die Behauptung.

Zu (ii): Sei also $R \subseteq M$ und gelte $\mathrm{R}\Sigma_j \mathrm{P}_{\mathcal{N}} = \Sigma_j \mathrm{P}_{\mathcal{N}}$. Sei X ein beliebiges Problem aus $\Sigma_j \mathrm{P}_{\mathcal{M}}$. Dann gibt es Polynome $p_1(n), \ldots, p_j(n)$ und ein $Y \in \mathrm{P}_{\mathcal{M}}$, so daß für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \exists \bar{y_1} \in M^{p_1(|\bar{x}|)} \ldots Q_j \bar{y_j} \in M^{p_j(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in Y.$$

Somit gilt für alle $\bar{x} \in N^*$:

$$\bar{x} \in X \Leftrightarrow \exists \bar{y_1} \in M^{p_1(|\bar{x}|)} \ldots Q_j \bar{y_j} \in M^{p_j(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in M^* \text{ und } \bar{x}\bar{y} \in Y.$$

Da $M \in \Sigma_i \mathrm{P}_{\mathcal{N}}$ können wir wieder wie oben die i entsprechenden Blöcke von Variablen und die in polynomieller Zeit zu überprüfenden Bedingungen einbauen. Da $i < j$ erhalten

wir insgesamt, daß $X \in \Sigma_j \mathrm{P}_{\mathcal{N}}$ gilt, und somit $X \in \mathrm{R}\Sigma_j \mathrm{P}_{\mathcal{N}}$. Damit gibt es aber Polynome $q_1(n), \ldots, q_j(n)$ und ein $Y' \in \mathrm{P}_{\mathcal{N}}$, so daß für alle $\bar{x} \in N^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \exists \bar{y_1} \in R^{p_1(|\bar{x}|)} \ldots Q_j \bar{y_j} \in R^{p_j(|\bar{x}|)} \text{ mit } \bar{x}\bar{y} \in Y'.$$

Da $\mathcal{M} \subseteq \mathcal{N}$ gilt, ist die deterministische Turingmaschine im Sinne von $\mathcal{N}$, welche das Problem Y' in polynomieller Zeit entscheidet auch eine deterministische Turingmaschine im Sinne von $\mathcal{M}$. Mit $R \subseteq M$ folgt damit $X \in \mathrm{R}\Sigma_j \mathrm{P}_{\mathcal{M}}$. □

Mit Hilfe des Lemmas 4.3.3 lässt sich nun folgender Satz beweisen, aus dem wir die Eigenschaften für die Standardtypen gewinnen werden:

Satz 4.3.6 *Seien $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $M \subseteq N$. Seien $i, j \in \mathbb{N}$ mit $i < j$ und gelte*

- $\Sigma_i \mathrm{P}_{\mathcal{N}} \restriction M = \Sigma_i \mathrm{P}_{\mathcal{M}}$
- $\Sigma_j \mathrm{P}_{\mathcal{N}} = \mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}}$
- *es gibt ein bezüglich $\mathrm{P}_{\mathcal{N}}$-Reduktion $\Sigma_j \mathrm{P}_{\mathcal{N}}$-vollständiges Problem $X \subseteq M^*$.*

Dann gilt:

(i) $\mathrm{PH}_{\mathcal{N}} = \Sigma_j \mathrm{P}_{\mathcal{N}} \Leftrightarrow \mathrm{PH}_{\mathcal{M}} = \Sigma_j \mathrm{P}_{\mathcal{M}}$

(ii) $\mathrm{PH}_{\mathcal{N}} = \Sigma_i \mathrm{P}_{\mathcal{N}} \Leftrightarrow \mathrm{PH}_{\mathcal{M}} = \Sigma_i \mathrm{P}_{\mathcal{M}}$

Beweis: Seien also $\mathcal{M}$ und $\mathcal{N}$ Strukturen und $i, j \in \mathbb{N}$ mit $i < j$ und den geforderten Eigenschaften.

Zu (i):

„$\rightarrow$“: Gelte $\mathrm{PH}_{\mathcal{N}} = \Sigma_i \mathrm{P}_{\mathcal{N}}$. Nach Lemma 4.3.3 gilt $\Pi_j \mathrm{P}_{\mathcal{M}} = \mathrm{M}\Pi_j \mathrm{P}_{\mathcal{N}} \restriction M$. Da nach Voraussetzung des Satzes $\Sigma_j \mathrm{P}_{\mathcal{N}} = \mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}}$ gilt, gilt auch $co\Sigma_j \mathrm{P}_{\mathcal{N}} = co\mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}}$ und somit:

$$\Pi_j \mathrm{P}_{\mathcal{M}} = \Pi_j \mathrm{P}_{\mathcal{N}} \restriction M.$$

Da nach Voraussetzung die polynomielle Hierarchie im Sinne von $\mathcal{N}$ auf die j-te Ebene kollabiert, gilt $\Pi_j \mathrm{P}_{\mathcal{N}} = \Sigma_j \mathrm{P}_{\mathcal{N}}$ und somit auch $\Pi_j \mathrm{P}_{\mathcal{N}} \restriction M = \Sigma_j \mathrm{P}_{\mathcal{N}} \restriction M$. Die Klasse $\Sigma_j \mathrm{P}_{\mathcal{N}} \restriction M$ ist nach Voraussetzung des Satzes gerade die Klasse $\mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}} \restriction M$. Dies ist aber mit Lemma 4.3.3 gerade die Klasse $\Sigma_j \mathrm{P}_{\mathcal{M}}$. Insgesamt ergibt sich also die Gleichungskette

$$\Pi_j \mathrm{P}_{\mathcal{M}} = \mathrm{M}\Pi_j \mathrm{P}_{\mathcal{N}} \restriction = \Pi_j \mathrm{P}_{\mathcal{N}} \restriction M = \Sigma_j \mathrm{P}_{\mathcal{N}} \restriction M = \mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}} \restriction M = \Sigma_j \mathrm{P}_{\mathcal{M}}$$

und die polynomielle Hierarchie im Sinne von $\mathcal{M}$ kollabiert mit Korollar 4.1.12 auf die j-te Ebene.

„$\leftarrow$“: Gelte $\mathrm{PH}_{\mathcal{M}} = \Sigma_j \mathrm{P}_{\mathcal{M}}$. Sei X das nach Voraussetzung existierende bezüglich $\mathrm{P}_{\mathcal{N}}$-Reduktion $\Sigma_j \mathrm{P}_{\mathcal{N}}$-vollständige Problem mit $X \subseteq M^*$. Dann ist $X \in \Sigma_j \mathrm{P}_{\mathcal{N}} \restriction M$. Mit der Voraussetzung $\Sigma_j \mathrm{P}_{\mathcal{N}} = \mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}}$ ist X somit in der Klasse $\mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}} \restriction M$. Dies ist mit Lemma 4.3.3 gerade die Klasse $\Sigma_j \mathrm{P}_{\mathcal{M}}$. Da die polynomielle Hierarchie im Sinne von $\mathcal{M}$ auf die j-te Ebene kollabiert, gilt $\Sigma_j \mathrm{P}_{\mathcal{M}} = \Pi_j \mathrm{P}_{\mathcal{M}}$. Wieder mit Lemma 4.3.3 gilt

$\Pi_j \mathrm{P}_{\mathcal{M}} = \mathrm{M}\Pi_j \mathrm{P}_{\mathcal{N}} \upharpoonright M$, und nach der Voraussetzung $\Sigma_j \mathrm{P}_{\mathcal{N}} = \mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}}$ des Satzes gilt $\mathrm{M}\Pi_j \mathrm{P}_{\mathcal{N}} \upharpoonright M = \Pi_j \mathrm{P}_{\mathcal{N}} \upharpoonright M$. Insgesamt ist das Problem X somit in der Klasse

$$\Sigma_j \mathrm{P}_{\mathcal{N}} \upharpoonright M = \mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}} \upharpoonright M = \Sigma_j \mathrm{P}_{\mathcal{M}} = \Pi_j \mathrm{P}_{\mathcal{M}} = \mathrm{M}\Pi_j \mathrm{P}_{\mathcal{N}} \upharpoonright M = \Pi_j \mathrm{P}_{\mathcal{N}} \upharpoonright M.$$

Da X ein $\Sigma_j \mathrm{P}_{\mathcal{N}}$-vollständiges Problem war, gilt somit

$$\Sigma_j \mathrm{P}_{\mathcal{N}} \subseteq \Pi_j \mathrm{P}_{\mathcal{N}}$$

und die polynomielle Hierarchie im Sinne von $\mathcal{N}$ kollabiert mit Korollar 4.1.13 auf die j-te Ebene.

Zu (ii):

„$\rightarrow$“: Gelte $\mathrm{PH}_{\mathcal{N}} = \Sigma_i \mathrm{P}_{\mathcal{N}}$. Mit Lemma 4.3.3 gilt $\Sigma_j \mathrm{P}_{\mathcal{M}} = \mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}} \upharpoonright M$. Nach Voraussetzung des Satzes ist dies gerade die Klasse $\Sigma_j \mathrm{P}_{\mathcal{N}} \upharpoonright M$. Da die polynomielle Hierarchie im Sinne von $\mathcal{N}$ auf die i-te Ebene kollabiert und $i < j$ nach Voraussetzung, gilt $\Sigma_j \mathrm{P}_{\mathcal{N}} = \Sigma_i \mathrm{P}_{\mathcal{N}}$ und somit auch $\Sigma_j \mathrm{P}_{\mathcal{N}} \upharpoonright M = \Sigma_i \mathrm{P}_{\mathcal{N}} \upharpoonright M$. Dies ist nach Voraussetzung des Satzes aber gerade die Klasse $\Sigma_i \mathrm{P}_{\mathcal{M}}$. Insgesamt gilt also

$$\Sigma_j \mathrm{P}_{\mathcal{M}} = \mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}} \upharpoonright M = \Sigma_j \mathrm{P}_{\mathcal{N}} \upharpoonright M = \Sigma_i \mathrm{P}_{\mathcal{N}} \upharpoonright M = \Sigma_i \mathrm{P}_{\mathcal{M}}$$

und mit Korollar 4.1.13 kollabiert die polynomielle Hierarchie im Sinne von $\mathcal{M}$ auf die i-te Ebene.

“$\leftarrow$“: Gelte $\mathrm{PH}_{\mathcal{M}} = \Sigma_i \mathrm{P}_{\mathcal{M}}$. Sei $X \subseteq M^*$ wieder das nach Voraussetzung existierende bezüglich $\mathrm{NP}_{\mathcal{N}}$-Reduktion $\Sigma_j \mathrm{P}_{\mathcal{N}}$-vollständige Problem. Das Problem X ist in der Klasse $\Sigma_j \mathrm{P}_{\mathcal{N}} \upharpoonright M$ und somit nach der Voraussetzung des Satzes in der Klasse $\mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}} \upharpoonright M$. Mit Lemma 4.3.3 ist X somit in $\Sigma_j \mathrm{P}_{\mathcal{M}}$. Da die polynomielle Hierarchie im Sinne von $\mathcal{M}$ auf die i-te Ebene kollabiert, gilt $\Sigma_j \mathrm{P}_{\mathcal{M}} = \Sigma_i \mathrm{P}_{\mathcal{M}}$, und X ist in $\Sigma_i \mathrm{P}_{\mathcal{M}}$. Dies ist allerdings nach Voraussetzung des Satzes gerade die Klasse $\Sigma_i \mathrm{P}_{\mathcal{N}} \upharpoonright M$. Das Problem X ist somit $\Sigma_j \mathrm{P}_{\mathcal{N}}$-vollständig und in der Klasse $\Sigma_i \mathrm{P}_{\mathcal{N}}$. Somit gilt $\Sigma_j \mathrm{P}_{\mathcal{N}} \subseteq \Sigma_i \mathrm{P}_{\mathcal{N}}$ und mit Korollar 4.1.13 kollabiert die polynomielle Hierarchie im Sinne von $\mathcal{N}$ auf die i-te Ebene. □

Da die Existenz eines $\Sigma_j \mathrm{P}_{\mathcal{N}}$-vollständigen Problems nur für die eine Richtung wichtig war, folgt aus dem Beweis des Satzes sofort:

Korollar 4.3.7 *Seien $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $M \subseteq N$. Seien $i, j \in \mathbb{N}$ mit $i < j$ und gelte*

- $\Sigma_i \mathrm{P}_{\mathcal{N}} \upharpoonright M = \Sigma_i \mathrm{P}_{\mathcal{M}}$
- $\Sigma_j \mathrm{P}_{\mathcal{N}} = \mathrm{M}\Sigma_j \mathrm{P}_{\mathcal{N}}$

Dann gilt:

(i) $\mathrm{PH}_{\mathcal{N}} = \Sigma_j \mathrm{P}_{\mathcal{N}} \Rightarrow \mathrm{PH}_{\mathcal{M}} = \Sigma_j \mathrm{P}_{\mathcal{M}}$

(ii) $\mathrm{PH}_{\mathcal{N}} = \Sigma_i \mathrm{P}_{\mathcal{N}} \Rightarrow \mathrm{PH}_{\mathcal{M}} = \Sigma_i \mathrm{P}_{\mathcal{M}}$

□

Mit diesem Satz können wir nun in dem H_i-Standardtyp Eigenschaften zusammenstellen, die dafür sorgen, daß sich die betreffende Struktur bezüglich eines Kollaps der polynomiellen Hierarchie ähnlich wie die Standardstruktur verhält:

Definition 4.3.8 (H_i-Standardtyp) Sei $\mathcal{M}$ eine Struktur und $i \in \mathbb{N}_{>0}$. Die Struktur $\mathcal{M}$ ist vom *H_i-Standardtyp*, falls gilt:

- $\mathrm{P}_{\mathcal{M}} \restriction \{0,1\} = \mathrm{P}$
- $\Sigma_i \mathrm{P}_{\mathcal{M}} = \mathrm{B}\Sigma_i \mathrm{P}_{\mathcal{M}}$
- es gibt ein boolesches bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\Sigma_i \mathrm{P}_{\mathcal{M}}$-vollständiges Problem.

Für Strukturen vom H_i-Standardtyp ist dann in der Tat die Frage, ob die polynomielle Hierarchie auf die i-te oder die 0-te Ebene kollabiert, äquivalent zu der Frage, ob die polynomielle Hierarchie über der Standardstruktur auf die i-te beziehungsweise die 0-te Ebene kollabiert:

Korollar 4.3.9 *Sei $\mathcal{M}$ eine Struktur vom H_i-Standardtyp. Dann gilt:*

(i) $\mathrm{PH}_{\mathcal{M}} = \Sigma_i \mathrm{P}_{\mathcal{M}} \Leftrightarrow \mathrm{PH} = \Sigma_i \mathrm{P}$

(ii) $\mathrm{PH}_{\mathcal{M}} = \mathrm{P}_{\mathcal{M}} \Leftrightarrow \mathrm{PH} = \mathrm{P}$.

Beweis: Sei $\mathcal{M}$ vom H_i-Standardtyp. Mit Satz 4.3.6 folgt sofort die Behauptung. □

Wir werden in Abschnitt 5.2 mit der Struktur der linearen Wörterbücher eine Struktur vom H_1-Standardtyp kennenlernen.

Generell haben Strukturen, in denen $\mathrm{P}_{\mathcal{M}} \restriction \{0,1\} = \mathrm{P}$ gilt, eine enge Verbindung zur Standardstruktur:

Bemerkung 4.3.10 Für eine Struktur $\mathcal{M}$ mit $\mathrm{P}_{\mathcal{M}} \restriction \{0,1\} = \mathrm{P}$ gilt für alle $i \in \mathbb{N}$:

$$\mathrm{B}\Sigma_i \mathrm{P}_{\mathcal{M}} \restriction \{0,1\} = \Sigma_i \mathrm{P}.$$

Für Fragen betreffend die nicht uniforme polynomielle Hierarchie über der Standardstruktur ist der $\mathbb{H}_i$-Standardtyp interessant:

Definition 4.3.11 ($\mathbb{H}_i$-Standardtyp) Sei $\mathcal{M}$ eine Struktur und $i \in \mathbb{N}_{>0}$. Die Struktur $\mathcal{M}$ ist vom *$\mathbb{H}_i$-Standardtyp*, falls gilt:

- $\mathrm{P}_{\mathcal{M}} \restriction \{0,1\} = \mathbb{P}$
- $\Sigma_i \mathrm{P}_{\mathcal{M}} = \mathrm{B}\Sigma_i \mathrm{P}_{\mathcal{M}}$
- es gibt ein boolesches bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\Sigma_i \mathrm{P}_{\mathcal{M}}$-vollständiges Problem.

Analog zum uniformen Fall gilt dann:

Satz 4.3.12 *Sei $\mathcal{M}$ eine Struktur vom $\mathbb{H}_i$-Standardtyp. Dann gilt:*

(i) $\mathrm{PH}_{\mathcal{M}} = \Sigma_i \mathrm{P}_{\mathcal{M}} \Leftrightarrow \mathbb{PH} = \Sigma_i \mathbb{P}$

(ii) $\mathrm{PH}_{\mathcal{M}} = \mathbb{P}_{\mathcal{M}} \Leftrightarrow \mathbb{PH} = \mathbb{P}$

Beweis: Analog zum Beweis von Satz 4.3.6. □

Allerdings sind in Strukturen mit höchstens abzählbarer Symbolmenge nur abzählbar viele Probleme in der Klasse $\mathrm{P}_{\mathcal{M}}$, während die Klasse $\mathbb{P}$ überabzählbar viele Probleme enthält. Der Begriff des $\mathbb{H}_i$-Standardtyps ist also nur für Strukturen mit genügend Konstanten interessant.

4.4 Ein Transfertheorem

Bevor wir uns den Beispielen zuwenden, werden wir nun ein Transfertheorem beweisen, welches besagt, daß sich elementar äquivalente Strukturen, also Strukturen, welche dieselben Formeln ohne freie Variablen erfüllen, bezüglich eines Kollaps' der polynomiellen Hierarchie gleich verhalten. Das Transfertheorem wurde zunächst von A. Hemmerling in [16] mit Hilfe von Berechnungsbäumen bewiesen. Wir werden allerdings etwas anders vorgehen: unser Beweis beruht auf der von Moritz Müller aufgezeigten Tatsache, daß sich zwei Strukturen $\mathcal{M}$ und $\mathcal{N}$, welche eine gemeinsame elementare Oberstruktur $\mathcal{L}$ besitzen, in bezug auf die $\mathrm{P}_{\mathcal{M}} = \mathrm{NP}_{\mathcal{M}}$-Frage gleich verhalten, falls es in der gemeinsamen Oberstruktur $\mathcal{L}$ ein slicewise definierbares und bezüglich $\mathrm{P}_{\mathcal{L}}$-Reduktion $\mathrm{NP}_{\mathcal{L}}$-vollständiges Problem gibt, welches eingeschränkt auf die Universen von $\mathcal{M}$ beziehungsweise $\mathcal{N}$ bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\mathrm{NP}_{\mathcal{M}}$-vollständig beziehungsweise bezüglich $\mathrm{P}_{\mathcal{N}}$-Reduktion $\mathrm{NP}_{\mathcal{N}}$-vollständig ist. Dies ist eine Verallgemeinerung und leichte Modifizierung des Beweises des Transfertheorems für algebraisch abgeschlossene Körper der Charakteristik 0 aus [3].

Wir benötigen zunächst den Begriff einer elementaren Substruktur.

Definition 4.4.1 (elementare Substruktur) Für zwei Strukturen $\mathcal{M}$ und $\mathcal{N}$ heißt die Struktur $\mathcal{M}$ *elementare Substruktur von* $\mathcal{N}$, falls $\mathcal{M}$ eine Substruktur von $\mathcal{N}$ ist, und falls für alle natürlichen Zahlen n, für alle Formeln $\varphi(\bar{x})$ mit n freien Variablen und für alle Wörter $\bar{a}$ aus M^n gilt:

$$\mathcal{M} \models \varphi(\bar{a}) \Leftrightarrow \mathcal{N} \models \varphi(\bar{a}).$$

Ist $\mathcal{M}$ eine elementare Substruktur von $\mathcal{N}$, so schreiben wir $\mathcal{M} \preceq \mathcal{N}$ und nennen $\mathcal{N}$ auch eine *elementare Oberstruktur* von $\mathcal{M}$.

Wir werden das Transfertheorem zunächst für Strukturen $\mathcal{M}$ und $\mathcal{N}$ mit $\mathcal{M} \preceq \mathcal{N}$ zeigen. Das gesuchte Ergebnis werden wir dann mit Hilfe eines Resultates aus der Modelltheorie erhalten. Betrachten wir Probleme aus einer Klasse $\Sigma_i \mathrm{P}_{\mathcal{N}}$ der polynomiellen Hierarchie im Sinne von $\mathcal{N}$. Diese haben einige nützliche Eigenschaften.

Lemma 4.4.2 *Seien $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $\mathcal{M} \preceq \mathcal{N}$. Dann gilt für jede postitive natürliche Zahl i und für jedes Problem $X \subseteq N^*$:*

$$X \in \Sigma_i \mathrm{P}_{\mathcal{N}} \Rightarrow X \cap M^* \in \Sigma_i \mathrm{P}_{\mathcal{M}}.$$

Beweis: Sei also $\mathcal{M} \preceq \mathcal{N}$ und gelte $X \in \Sigma_i \mathrm{P}_{\mathcal{N}}$. Dann gibt es eine in polynomieller Zeit berechenbare gesetzte Schaltkreisfamilie $(C_n)_{n\in\mathbb{N}_{>0}}$ im Sinne von $\mathcal{N}$ und Polynome $p_1, \ldots, p_i$, so daß für alle $\bar{x} \in N^*$ gilt:

$$\bar{x} \in X \Leftrightarrow \exists \bar{y}_1 \in N^{p_1(|\bar{x}|)} \ldots Q_i \bar{y}_i \in N^{p_i(|\bar{x}|)} \text{ mit } C_{|\bar{x}|+p_1(|\bar{x}|)+\ldots+p_i(|\bar{x}|)}(\bar{x}, \bar{y}_1, \ldots, \bar{y}_i) = 1.$$

Mit Bemerkung 1.1.4 gibt es andererseits eine Familie von Formeln $(\psi_n)_{n\in\mathbb{N}_{>0}}$, so daß für alle $\bar{x} \in N^*$ gilt:

$$C_{|\bar{x}|}(\bar{x}) = 1 \Leftrightarrow \mathcal{N} \models \psi_{|\bar{x}|}(\bar{x}).$$

Wir schreiben $p(|\bar{x}|)$ für $|\bar{x}| + p_1(|\bar{x}|) + \ldots + p_i(|\bar{x}|)$ und $\bar{y}$ für $\bar{y}_1, \ldots, \bar{y}_i$ und erhalten für alle $\bar{x}$ aus N^*:

$$\begin{aligned} \bar{x} \in X &\Leftrightarrow \exists \bar{y}_1 \in N^{p_1(|\bar{x}|)} \ldots Q_i \in N^{p_i(|\bar{x}|)} \bar{y}_i \text{ mit } \mathcal{N} \models \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y}) \\ &\Leftrightarrow \mathcal{N} \models \exists \bar{y}_1 \ldots Q_i \bar{y}_i \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y}) \end{aligned}$$

Da $\mathcal{M}$ eine Substruktur von $\mathcal{N}$ ist, und deshalb $M \subseteq N$ gilt, gilt somit aber im Speziellen auch für alle $\bar{x}$ aus M^*:

$$\bar{x} \in X \cap M^* \Leftrightarrow \mathcal{N} \models \exists \bar{y}_1 \ldots Q_i \bar{y}_i \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y}).$$

Da $\mathcal{M}$ eine elementare Substruktur von $\mathcal{N}$ ist, gilt damit für alle $\bar{x} \in M^*$ auch:

$$\bar{x} \in X \cap M^* \Leftrightarrow \mathcal{M} \models \exists \bar{y}_i \ldots Q_i \bar{y}_i \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y})$$

und aufgrund der Definition der Formeln $(\psi_n)_{n\in\mathbb{N}_{>0}}$ gilt für alle $\bar{x} \in M^*$ somit:

$$\bar{x} \in X \cap M^* \Leftrightarrow \exists \bar{y}_i \in M^{p_1(|\bar{x}|)} \ldots Q_i \bar{y}_i \in M^{p_i(|\bar{x}|)} \text{ mit } C_{p(|\bar{x}|)}(\bar{x}, \bar{y}) = 1.$$

Da die Schaltkreise C_n in klassischem Sinne und somit auch im Sinne der Struktur $\mathcal{M}$ in polynomieller Zeit aus n berechenbar sind, läßt sich die Bedingung $C_{p(|\bar{x}|)}(\bar{x}, \bar{y}) = 1$ im Sinne von $\mathcal{M}$ in polynomieller Zeit überprüfen, und es gilt $X \cap M^* \in \Sigma_i \mathrm{P}_{\mathcal{M}}$. □

Desweiteren sind auch die Restriktionen von bezüglich $\mathrm{P}_{\mathcal{N}}$-Reduktion $\Sigma_i \mathrm{P}_{\mathcal{N}}$-vollständigen Problemen auf das Universum einer elementaren Substruktur $\mathcal{M}$ von $\mathcal{N}$ bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\Sigma_i \mathrm{P}_{\mathcal{M}}$-vollständig:

Lemma 4.4.3 *Seien $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $\mathcal{M} \preceq \mathcal{N}$. Für eine natürliche Zahl i sei $X \subseteq N^*$ ein bezüglich $\mathrm{P}_{\mathcal{N}}$-Reduktion $\Sigma_i \mathrm{P}_{\mathcal{N}}$-vollständiges Problem. Dann ist $X \cap M^*$ ein bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\Sigma_i \mathrm{P}_{\mathcal{M}}$-vollständiges Problem.*

Beweis: Seien also $\mathcal{M}, \mathcal{N}$ wie gefordert und sei $X \subseteq N^*$ für eine natürliche Zahl i ein bezüglich $\mathrm{P}_{\mathcal{N}}$-Reduktion $\Sigma_i \mathrm{P}_{\mathcal{N}}$-vollständiges Problem. Dann gilt zunächst mit dem vorhergehenden Lemma 4.4.2, daß das Problem $X \cap M^*$ in der Klasse $\Sigma_i \mathrm{P}_{\mathcal{M}}$ liegt. Wir müssen also noch zeigen, daß sich jedes andere Problem in $\Sigma_i \mathrm{P}_{\mathcal{M}}$ im Sinne von $\mathcal{M}$ in polynomieller Zeit auf $X \cap M^*$ reduzieren läßt.

Sei dazu Y ein Problem aus $\Sigma_i \mathrm{P}_{\mathcal{M}}$. Dann gibt es wieder eine im klassischen Sinne in polynomieller Zeit berechenbare gesetzte Familie von Schaltkreisesn $(C_n)_{n\in\mathbb{N}_{>0}}$ und

Polynome $p_1, \ldots, p_i$, so daß mit den Schreibweisen $p(n)$ für $n + p_1(n) + \ldots + p_i(n)$ und $\bar{y}$ für $\bar{y_1}, \ldots, \bar{y_i}$ für alle Wörter $\bar{x}$ aus M^* gilt:

$$\bar{x} \in Y \Leftrightarrow \exists y_1 \in M^{p_1(|\bar{x}|)} \ldots Q_i y_i \in M^{p_i(|\bar{x}|)} \text{ mit } C_{p(|\bar{x}|)}(\bar{x}, \bar{y}) = 1.$$

Mit Bemerkung 1.1.4 gibt es wieder eine Familie von Formeln $(\psi_n)_{n \in \mathbb{N}_{>0}}$, so daß für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in Y \Leftrightarrow \mathcal{M} \models \exists \bar{y_1} \ldots Q_i \bar{y_i} \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y}).$$

Da $\mathcal{M}$ elementare Substruktur von $\mathcal{N}$ ist, gilt wieder für alle $\bar{x} \in M^*$:

$$\bar{x} \in Y \Leftrightarrow \mathcal{N} \models \exists \bar{y_1} \ldots Q_i \bar{y_i} \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y})$$

und aufgrund der Definition der Formeln ψ_n gilt somit für alle $\bar{x}$ aus M^*:

$$\bar{x} \in Y \Leftrightarrow \exists \bar{y_1} \in N^{p_1(|\bar{x}|)} \ldots Q_i \bar{y_i} \in N^{p_i(|\bar{x}|)} \text{ mit } C_{p(|\bar{x}|)}(\bar{x}, \bar{y}) = 1.$$

Wir definieren nun ein Problem $\tilde{Y} \subseteq N^*$ durch:

$$\tilde{Y} := \{\bar{x} \in N^* \mid \exists \bar{y_1} \in N^{p_1(|\bar{x}|)} \ldots Q_i \bar{y_i} \in N^{p_i(|\bar{x}|)} \text{ mit } C_{p(|\bar{x}|)}(\bar{x}, \bar{y}) = 1\}.$$

Dann gilt $\tilde{Y} \cap M^* = Y$. Da Schaltkreise im Sinne von $\mathcal{M}$ auch Schaltkreise im Sinne von $\mathcal{N}$ sind, ist $\tilde{Y}$ in der Klasse $\Sigma_i \mathrm{P}_{\mathcal{N}}$. Da nach Voraussetzung das Problem X bezüglich $\mathrm{P}_{\mathcal{N}}$-Reduktion $\Sigma_i \mathrm{P}_{\mathcal{N}}$-vollständig ist, gibt es also eine $\mathrm{P}_{\mathcal{N}}$-Reduktion $f : N^* \to N^*$ von $\tilde{Y}$ auf X. Der die Funktion f in polynomieller Zeit im Sinne von $\mathcal{N}$ berechnende Algorithmus $\mathbb{A}$ ist, da $\mathcal{M}$ eine Substruktur von $\mathcal{N}$ ist, auch ein Algorithmus im Sinne von $\mathcal{M}$ und berechnet bei Eingaben aus M^* Wörter aus M^*. Somit gilt aber für alle $\bar{x}$ aus M^*:

$$\begin{aligned} \bar{x} \in Y &\Leftrightarrow \bar{x} \in \tilde{Y} \cap M^* \\ &\Leftrightarrow f(\bar{x}) \in X \cap M^* \end{aligned}$$

Somit ist die Funktion f eingeschränkt auf M^* eine $\mathrm{P}_{\mathcal{M}}$-Reduktion von Y auf das Problem $X \cap M^*$. Da die Konstruktion der Funktion f für jedes Problem Y aus $\Sigma_i \mathrm{P}_{\mathcal{M}}$ möglich ist, ist $X \cap M^*$ ein bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\Sigma_i \mathrm{P}_{\mathcal{M}}$-vollständiges Problem. □

Da wir die Eigenschaft, daß eine elementare Substruktur einer Struktur die gleichen Formeln erfüllt, wie die Struktur selber, ausnützen wollen, ist die folgende Beobachtung nützlich.

Lemma 4.4.4 *Sei $\mathcal{M}$ eine Struktur und i eine natürliche Zahl. Dann gibt es für jedes Problem X aus $\Sigma_i \mathrm{P}_{\mathcal{M}}$ eine Familie $(\varphi_n)_{n \in \mathbb{N}_{>0}}$ von Formeln der Sprache von $\mathcal{M}$, so daß für jede postitive natürliche Zahl n die Formel φ_n gerade n freie Variablen besitzt, und so daß gilt:*

$$X = \{\bar{x} \in M^* \mid \mathcal{M} \models \varphi_{|\bar{x}|}(\bar{x})\}.$$

Beweis: Sei X also ein Problem aus $\Sigma_i \mathrm{P}_{\mathcal{M}}$. Dann gibt es eine in polynomieller Zeit berechenbare gesetzte Familie $(C_n)_{n \in \mathbb{N}_{>0}}$ im Sinne von $\mathcal{M}$ und Polynome $p_1, \ldots, p_i$, so

daß (wieder mit den Notationen $p(n)$ für $n + p_1(n) + \ldots + p_i(n)$ und $\bar{y}$ für $\bar{y_1}, \ldots, \bar{y_i}$) für für alle Wörter $\bar{x}$ aus M^* gilt:

$$\bar{x} \in X \Leftrightarrow \exists \bar{y_1} \in M^{p_1(|\bar{x}|)} \ldots Q_i \bar{y_i} \in M^{p_i(\bar{x})} \text{ mit } C_{p(|\bar{x}|)}(\bar{x}, \bar{y}) = 1.$$

Mit Bemerkung 1.1.4 gibt es nun wieder eine Familie $(\psi_n)_{n \in \mathbb{N}_{>0}}$ von Formeln, so daß für alle $\bar{x}$ aus M^* gilt:

$$\begin{aligned} \bar{x} \in X &\Leftrightarrow \exists \bar{y_1} \in M^{p_1(|\bar{x}|)} \ldots Q_i \bar{y_i} \in M^{p_i(\bar{x})} \text{ mit } \mathcal{M} \models \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y}) \\ &\Leftrightarrow \mathcal{M} \models \exists \bar{y_1} \ldots Q_i \bar{y_i} \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y}) \end{aligned}$$

Somit gilt für die Familie $(\varphi_n)_{n \in \mathbb{N}_{>0}}$ von Formeln mit $\varphi_n(\bar{x}) := \exists \bar{y_1} \ldots Q_i \bar{y_i} \psi_{p(n)}(\bar{x}, \bar{y})$:

$$X = \{\bar{x} \in M^* \mid \mathcal{M} \models \varphi_{|\bar{x}|}(\bar{x})\}.$$

□

Mit einer solchen Familie von Formeln läßt sich dann auch die Restriktion eines Problems auf das Universum einer elementaren Substruktur definieren.

Lemma 4.4.5 *Seien $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $\mathcal{M} \preceq \mathcal{N}$ und sei für eine Familie $(\varphi_n)_{n \in \mathbb{N}_{>0}}$ von Formeln der Sprache von $\mathcal{N}$ ein Problem $X \subseteq N^*$ definiert durch $X = \{\bar{x} \in N^* \mid \mathcal{N} \models \varphi_{|\bar{x}|}(\bar{x})\}$. Dann gilt*

$$X \cap M^* = \{\bar{x} \in M^* \mid \mathcal{M} \models \varphi_{|\bar{x}|}(\bar{x})\}.$$

Beweis: Gelte also $X = \{\bar{x} \in N^* \mid \mathcal{N} \models \varphi_{|\bar{x}|}(\bar{x})\}$. Dann gilt aber

$$\begin{aligned} X \cap M^* &= \{\bar{x} \in N^* \mid \mathcal{N} \models \varphi_{|\bar{x}|}(\bar{x})\} \cap M^* \\ &= \{\bar{x} \in M^* \mid \mathcal{N} \models \varphi_{|\bar{x}|}(\bar{x})\} \end{aligned}$$

Dies ist aber, da $\mathcal{M}$ eine elementare Substruktur von $\mathcal{N}$ ist, gerade die Menge $\{\bar{x} \in M^* \mid \mathcal{M} \models \varphi_{|\bar{x}|}(\bar{x})\}$, woraus die Behauptung folgt. □

Mit diesen Hilfsmitteln können wir nun das angekündigte Transfertheorem für elementare Substrukturen zeigen.

Satz 4.4.6 (Transfertheorem für elementare Substrukturen) *Seien $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $\mathcal{M} \preceq \mathcal{N}$. Dann gilt für alle natürlichen Zahlen i:*

$$\mathrm{PH}_{\mathcal{M}} = \Sigma_i \mathrm{P}_{\mathcal{M}} \Leftrightarrow \mathrm{PH}_{\mathcal{N}} = \Sigma_i \mathrm{P}_{\mathcal{N}}.$$

Beweis: Seien also $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $\mathcal{M} \preceq \mathcal{N}$ und i eine natürliche Zahl. Zunächst gibt es mit Bemerkung 4.2.8 generell ein bezüglich $\mathrm{P}_{\mathcal{N}}$-Reduktion $\Sigma_{i+1}\mathrm{P}_{\mathcal{N}}$-vollständiges Problem $X \subseteq N^*$. Mit Lemma 4.4.3 ist das Problem $X \cap M^*$ dann bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\Sigma_{i+1}\mathrm{P}_{\mathcal{M}}$-vollständig. Außerdem gibt es mit Lemma 4.4.4 eine Familie

$(\varphi_n)_{n\in\mathbb{N}_{>0}}$ von Formeln, so daß für jede positive natürliche Zahl n die Formel φ_n eine Formel mit n freien Variablen ist, und so daß gilt:

$$X = \{\bar{x} \in N^* \mid \mathcal{N} \models \varphi_{|\bar{x}|}(\bar{x})\}.$$

Mit Lemma 4.4.5 gilt zudem

$$X \cap M^* = \{\bar{x} \in M^* \mid \mathcal{M} \models \varphi_{|\bar{x}|}(\bar{x})\}.$$

Betrachten wir nun die einzelnen Implikationen.
„→“: Gelte also $\mathrm{PH}_{\mathcal{M}} = \Sigma_i \mathrm{P}_{\mathcal{M}}$. Dann ist also auch das Problem $X \cap M^*$ in der Klasse $\Sigma_i \mathrm{P}_{\mathcal{M}}$. Somit gibt es eine Familie $(C_n)_{n\in\mathbb{N}_{>0}}$ von gesetzten Schaltkreisen im Sinne von $\mathcal{M}$, welche in polynomieller Zeit berechenbar sind, und Polynome $p_1, \ldots, p_i$, so daß (wieder in abkürzender Schreibweise) für alle $\bar{x} \in M^*$ gilt:

$$\bar{x} \in X \cap M^* \Leftrightarrow \exists \bar{y_1} \in M^{p_1(|\bar{x}|)} \ldots Q_i \bar{y_i} \in M^{p_i(|\bar{x}|)} \text{ mit } C_{p(|\bar{x}|)}(\bar{x}, \bar{y}) = 1.$$

Es gibt also wieder eine Familie $(\psi_n)_{n\in\mathbb{N}_{>0}}$ von Formeln, so daß für alle $\bar{x}$ aus M^* gilt:

$$C_{|\bar{x}|}(\bar{x}) = 1 \Leftrightarrow \mathcal{M} \models \psi_{|\bar{x}|}(\bar{x}).$$

Somit gilt also auch für alle Wörter $\bar{x} \in M^*$:

$$\bar{x} \in X \cap M^* \Leftrightarrow \mathcal{M} \models \exists \bar{y_i} \ldots Q_i \bar{y_i} \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y}).$$

Mit der Vorüberlegung gilt andererseits $\bar{x} \in X \cap M^* \Leftrightarrow \mathcal{M} \models \varphi_{|\bar{x}|}(\bar{x})$. Deshalb gilt für alle $\bar{x} \in M^*$:

$$\mathcal{M} \models \varphi_{|\bar{x}|}(\bar{x}) \Leftrightarrow \mathcal{M} \models \exists \bar{y_i} \ldots Q_i \bar{y_i} \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y})$$

und somit auch

$$\mathcal{M} \models \forall \bar{x} \left(\varphi_{|\bar{x}|}(\bar{x}) \leftrightarrow \exists \bar{y_i} \ldots Q_i \bar{y_i} \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y}) \right).$$

Da $\mathcal{M}$ eine elementare Substruktur von $\mathcal{N}$ ist, gilt damit aber auch

$$\mathcal{N} \models \forall \bar{x} \left(\varphi_{|\bar{x}|}(\bar{x}) \leftrightarrow \exists \bar{y_i} \ldots Q_i \bar{y_i} \psi_{p(|\bar{x}|)}(\bar{x}, \bar{y}) \right)$$

und somit gilt mit der Definition der Formeln φ_n für alle $\bar{x}$ aus N^*:

$$\bar{x} \in X \Leftrightarrow \exists \bar{y_1} \in N^{p_1(|\bar{x}|)} \ldots Q_i \bar{y_i} \in N^{p_i(|\bar{x}|)} \text{ mit } C_{p(|\bar{x}|)}(\bar{x}, \bar{y}) = 1.$$

Da man Schaltkreise im Sinne von $\mathcal{N}$ in polynomieller Zeit auswerten kann, ist die Bedingung $C_{p(|\bar{x}|)}(\bar{x}, \bar{y}) = 1$ eine $\mathrm{P}_{\mathcal{N}}$-Bedingung. Somit gilt $X \in \Sigma_i \mathrm{P}_{\mathcal{N}}$, und die polynomielle Hierarchie im Sinne von $\mathcal{N}$ kollabiert auf die i-te Ebene.

„←“: Gelte andererseits $\mathrm{P}_{\mathcal{N}} = \Sigma_i \mathrm{P}_{\mathcal{N}}$. Dann ist das Problem X in der Klasse $\Sigma_i \mathrm{P}_{\mathcal{N}}$. Mit Lemma 4.4.2 ist damit aber auch das Problem $X \cap M^*$ in der Klasse $\Sigma_i \mathrm{P}_{\mathcal{M}}$. Da das Problem $X \cap M^*$ nach den Vorbemerkungen $\Sigma_{i+1} \mathrm{P}_{\mathcal{M}}$-vollständig bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion ist, gilt somit $\Sigma_{i+1} \mathrm{P}_{\mathcal{M}} = \Sigma_i \mathrm{P}_{\mathcal{M}}$, und die polynomielle Hierarchie im Sinne von $\mathcal{M}$ kollabiert auf die i-te Ebene. □

Wir wollen nun den Fall von Strukturen, in denen dieselben Formeln ohne freie Variablen gelten, betrachten.

Definition 4.4.7 Zwei Strukturen $\mathcal{M}$ und $\mathcal{N}$ mit gleicher Symbolmenge heißen *elementar äquivalent*, falls für jede Formel φ der Sprache von $\mathcal{N}$ ohne freie Variablen gilt:

$$\mathcal{M} \models \varphi \Leftrightarrow \mathcal{N} \models \varphi.$$

Sind $\mathcal{M}$ und $\mathcal{N}$ elementar äquivalent, so schreiben wir auch $\mathcal{M} \equiv \mathcal{N}$.

Wir werden ein Ergebnis aus der Modelltheorie verwenden, welches wir an dieser Stelle jedoch nicht beweisen werden. Für den Beweis siehe zum Beispiel [13], Seite 46 oder [23], ebenfalls Seite 46.

Satz 4.4.8 *Seien $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $\mathcal{M} \equiv \mathcal{N}$. Dann gibt es eine elementare Oberstruktur $\mathcal{L}$ von $\mathcal{M}$, so daß $\mathcal{N}$ isomorph zu einer elementaren Substruktur von $\mathcal{L}$ ist.*

Das angekündigte Transfertheorem ergibt sich nun widerstandslos.

Satz 4.4.9 (Transfertheorem für elementar äquivalente Strukturen) *Seien $\mathcal{M}$ und $\mathcal{N}$ Strukturen mit $\mathcal{M} \equiv \mathcal{N}$. Dann gilt für jede natürliche Zahl i:*

$$\mathrm{PH}_{\mathcal{M}} = \Sigma_i \mathrm{P}_{\mathcal{M}} \Leftrightarrow \mathrm{PH}_{\mathcal{N}} = \Sigma_i \mathrm{P}_{\mathcal{N}}.$$

Beweis: Wenn die beiden Strukturen $\mathcal{M}$ und $\mathcal{N}$ elementar äquivalent sind, so gibt es mit Satz 4.4.8 eine elementare Oberstruktur $\mathcal{L}$ von $\mathcal{M}$, so daß $\mathcal{N}$ isomorph zu einer elementaren Substruktur von $\mathcal{L}$ ist. Mit dem Transfertheorem für elementare Substrukturen und Bemerkung 4.1.8 gilt dann

$$\begin{aligned} \mathrm{PH}_{\mathcal{M}} = \Sigma_i \mathrm{P}_{\mathcal{M}} &\Leftrightarrow \mathrm{PH}_{\mathcal{L}} = \Sigma_i \mathrm{P}_{\mathcal{L}} \\ &\Leftrightarrow \mathrm{PH}_{\mathcal{N}} = \Sigma_i \mathrm{P}_{\mathcal{N}} \end{aligned}$$

und somit die Behauptung. □

Kapitel 5

Anwendungen / Beispiele

Wir wollen abschließend verschiedene Strukturen betrachten und in Hinblick darauf untersuchen, ob sie vom H_1-Standardtyp sind. Ist dies für eine Struktur $\mathcal{M}$ der Fall, so erinnern wir uns, daß die Frage, ob $\mathrm{P}_{\mathcal{M}} = \mathrm{NP}_{\mathcal{M}}$ gilt, äquivalent zu der Frage ist, ob $\mathrm{P} = \mathrm{NP}$ gilt. Zunächst werden wir uns mit Strukturen mit endlicher relationaler Symbolmenge beschäftigen. Dann werden wir mit der Struktur der linearen Wörterbücher eine Struktur kennenlernen, die vom H_1-Standardtyp ist. Im Anschluß an diese Überlegungen betrachten wir verschiedene Strukturen über den reellen Zahlen, und als letztes wenden wir uns Körpern zu. Die Ergebnisse dieses Kapitels sind im wesentlichen die Ergebnisse aus [22] beziehungsweise [26], bis auf die Ergebnisse in Lemma 5.2.1 und Satz 5.3.6, welche aufgrund des leicht modifizierten Berechnungsmodells dort nur in einer nicht uniformen Version gelten.

Um zu zeigen, daß eine Struktur vom H_1-Standardtyp ist, müssen wir unter anderem die Eigenschaft $\mathrm{P}_{\mathcal{M}} \restriction \{0,1\} = \mathrm{P}$ nachweisen. Falls $\mathcal{M}$ eine endliche Struktur ist, so gilt $\mathrm{P}_{\mathcal{M}} \restriction \{0,1\} = \mathrm{P}$ sofort, da die endlich vielen Elemente von M durch endliche 0-1-Folgen kodiert werden können. Eine Turingmaschine im Sinne von $\mathcal{M}$ kann somit durch eine klassische Turingmaschine simuliert werden, indem die Funktionen und Relationen von $\mathcal{M}$ durch Subroutinen über der Standardstruktur berechnet werden. Da die Symbolmenge von $\mathcal{M}$ nach unserer Generalvoraussetzung nur endlich viele Funktions- und Relationssymbole enthält, ist die Anzahl der von den Subroutinen benötigten Schritte beschränkt, und die insgesamt benötigte Zeit vergrößert sich nur um einen linearen Faktor.

Es gilt sogar:

Lemma 5.0.10 *Sei $\mathcal{M}$ eine Struktur, so daß die von $\{0,1\}$ erzeugte Substruktur endlich ist. Dann gilt:*

$$\mathrm{P}_{\mathcal{M}} \restriction \{0,1\} = \mathrm{P}.$$

Beweis: Sei also $\mathcal{M}$ eine Struktur mit $[\{0,1\}]^{\mathcal{M}}$ endlich. Für ein beliebiges $X \in \mathrm{P}_{\mathcal{M}} \restriction \{0,1\}$ konstruieren wir eine das Problem X in polynomieller Zeit entscheidende deterministische Turingmaschine $\mathbb{A}'$ im klassischen Sinne wie folgt:

Da die von $\{0,1\}$ erzeugte Substruktur in $\mathcal{M}$ endlich ist, können wir alle ihre Elemente durch endliche $0-1$-Folgen kodieren. Desweiteren konstruieren wir uns wie für eine endliche Struktur die Subroutinen, welche die Funktionen und Relationen von $\mathcal{M}$ auf den Kodierungen der Elemente von $[\{0,1\}]^{\mathcal{M}}$ simulieren. Da $X \in \mathrm{P}_{\mathcal{M}} \restriction \{0,1\}$ gilt, gibt es

eine Turingmaschine $\mathbb{A}$ im Sinne von $\mathcal{M}$, welche das Problem X für Eingaben aus $\{0,1\}^*$ in polynomieller Zeit entscheidet. Die von und konsruierte klassische Turingmaschine $\mathbb{A}'$ überprüft bei Eingabe von $\bar{x} \in M^*$ nun zuerst, ob $\bar{x} \in \{0,1\}^*$ gilt, und gibt 0 aus, falls dies nicht der Fall sein sollte. Andernfalls simuliert $\mathbb{A}'$ die Maschine $\mathbb{A}$, indem sie die Funktionen und Relationen von $\mathcal{M}$ mit den entsprechenden Subroutinen simuliert. Da das Überprüfen der Eingabe in linearer Zeit möglich ist, und da die von den Subroutinen benötigte Zeit beschränkt ist, vergrößert sich die benötigte Zeit wieder nur um einen linearen Faktor. Die klassische Turingmaschine $\mathbb{A}'$ entscheidet also X in polynomieller Zeit, es gilt $X \in \mathrm{P}$. □

Um zu zeigen, daß eine Struktur, deren von $\{0,1\}$ erzeugte Substruktur endlich ist, vom H_1-Standardtyp ist, genügt es also, die Gleichheit von $\mathrm{NP}_{\mathcal{M}}$ und $\mathrm{BNP}_{\mathcal{M}}$ und die Existenz eines booleschen $\mathrm{NP}_{\mathcal{M}}$-vollständigen Problems zu zeigen. Es wird noch angenehmer, wenn wir Strukturen mit endlicher relationaler Symbolmenge betrachten:

5.1 Strukturen mit endlicher relationaler Symbolmenge

Da in Strukturen mit endlicher relationaler Symbolmenge, also in Strukturen, deren Symbolmenge (außer dem Selektor und der Identität) nur endlich viele Konstanten- und Relationssymbole enthält, die von $\{0,1\}$ erzeugte Substruktur endlich ist, gilt mit Lemma 5.0.10 die Gleichheit von $\mathrm{P}_{\mathcal{M}} \restriction \{0,1\}$ und P. Wir wollen Kriterien dafür angeben, daß eine solche Struktur vom H_1-Standardtyp ist.

Wir werden sehen, daß es in jeder Struktur mit endlicher relationaler Symbolmenge, welche Quantorenelimination erlaubt, ein boolesches $\mathrm{NP}_{\mathcal{M}}$-vollständiges Problem gibt, das Problem der Erfüllbarkeit quantorenfreier Formeln ohne Parameter:

Sat(Qf Fml ohne Par in $\mathcal{M}$)
Eingabe: Eine quantorenfreie Formel $\varphi(\bar{x})$ in binärer Kodierung
Frage: Gibt es ein $\bar{a} \in M^{\lvert\bar{x}\rvert}$ mit $\mathcal{M} \models \varphi(\bar{a})$?

Um dies zu zeigen, benötigen wir den Begriff des quantorenfreien Typs eines Elementes aus M^*.

Definition 5.1.1 (quantorenfreier Typ) Sei $\mathcal{M}$ eine Struktur mit endlicher relationaler Symbolmenge. Für ein $\bar{x} \in M^*$ ist der *quantorenfreie Typ von* $\bar{x}$ die Konjunktion der atomaren oder negiert atomaren Formeln in $|\bar{a}|$ Variablen, die von $\bar{a}$ erfüllt werden:

$$\text{qf-tp}_{\mathcal{M}}[\bar{a}] := \bigwedge_{\psi \text{ Literal, } \mathcal{M} \models \psi(\bar{a})} \psi(x_1, \ldots, x_{|\bar{a}|})$$

Die Reihenfolge der Literale in der Konjunktion ist dabei z.B. durch die lexikographische Ordnung und die Ordnung der Relations- und Konstantensymbole festgelegt. Da die Symbolmenge von $\mathcal{M}$ relational und endlich ist, ist $\text{qf-tp}_{\mathcal{M}}[\bar{a}]$ eine endliche Konjunktion.

Da außerdem die Symbolmenge jeder der von uns betrachteten Strukturen die Konstantensymbole 0 und 1, die Funktionssymbole Id und S und das Relationssymbol $=$ enthält, ist der quantorenfreie Typ auch für das leere Wort definiert.

Der quantorenfreie Typ hat einige nützliche Eigenschaften:

Lemma 5.1.2 (Eigenschaften von qf-tp$_\mathcal{M}$**)** *Sei $\mathcal{M}$ eine Struktur mit endlicher relationaler Symbolmenge und sei $\bar{a} \in M^*$. Dann gilt für alle quantorenfreien Formeln φ in $|\bar{a}|$ freien Variablen:*

(i) qf-tp$_\mathcal{M}[\bar{a}]$ *kann binär kodiert werden*

(ii) Die Funktion $f : M^* \to \{0,1\}^*$ mit $f(\bar{x}) :=$ qf-tp$_\mathcal{M}[\bar{x}]$ *ist in polynomieller Zeit im Sinne von $\mathcal{M}$ berechenbar*

(iii) $\mathcal{M} \models \varphi(\bar{a})$ genau dann, wenn φ aus qf-tp$_\mathcal{M}[\bar{a}]$ *beweisbar ist.*

Beweis: *Zu (i):* Klar, da die Symbolmenge von $\mathcal{M}$ endlich ist.

Zu (ii): Wir konstruieren einen Algorithmus, der die Funktion f in polynomieller Zeit im Sinne von $\mathcal{M}$ berechnet wie folgt: Bei Eingabe von $\bar{x} \in M^*$ geht der Algorithmus der Reihe nach alle möglichen Relationen und Kombinationen von $x_1, \ldots, x_{|\bar{x}|}$ durch, und prüft, ob die Relationen auf die entsprechenden Kombinationen zutreffen. Ist dies der Fall, so schreibt er die entsprechende atomare Formel zu dem quantorenfreien Typ dazu, andernfalls ihre Negation. Die benötigte Laufzeit ist zwar exponentiell in der maximalen Stelligkeit der Relationssymbole, aber nur polynomiell in der Länge von $\bar{x}$.

Zu (iii): Wenn φ eine quantorenfreie Formel in $|\bar{a}|$ Variablen ist, so ist φ eine boolesche Kombination von atomaren Formeln in $|\bar{a}|$ Variablen. Da in qf-tp$_\mathcal{M}[\bar{a}]$ festgelegt ist, welche der atomaren Formeln von $\bar{a}$ erfüllt werden, ist φ also genau dann aus qf-tp$_\mathcal{M}[\bar{a}]$ beweisbar, falls $\varphi(\bar{a})$ in $\mathcal{M}$ gilt. □

Mit Hilfe des quantorenfreien Typs können wir nun das Problem Sat(Qf Fml mit Par in $\mathcal{M}$) auf das Problem Sat(Qf Fml ohne Par in $\mathcal{M}$) reduzieren, und somit die NP$_\mathcal{M}$-Vollständigkeit von Sat(Qf Fml ohne Par in $\mathcal{M}$) beweisen:

Lemma 5.1.3 (satqf-p npm-vollst) *Sei $\mathcal{M}$ eine Struktur mit endlicher relationaler Symbolmenge, welche Quantorenelimination erlaubt. Dann ist das Problem* Sat(Qf Fml ohne Par in $\mathcal{M}$) *bezüglich* P$_\mathcal{M}$*-Reduktion* NP$_\mathcal{M}$*-vollständig und bezüglich* $\mathbb{P}_\mathcal{M}$*-Reduktion* $\mathbb{NP}_\mathcal{M}$*-vollständig.*

Beweis: Wir beweisen den uniformen Fall, der nichtuniforme Fall folgt dann sofort. Zunächst ist das Problem Sat(Qf Fml ohne Par in $\mathcal{M}$) in der Klasse NP$_\mathcal{M}$, denn eine nichtdeterministische Turingmaschine im Sinne von $\mathcal{M}$, welche bei Eingabe einer quantorenfreien Formel in binärer Kodierung $\ulcorner\varphi(x_1, \ldots, x_n)\urcorner$ ein Wort $\bar{a} \in M^{|\ulcorner\varphi\urcorner|}$ rät und (in polynomieller Zeit) prüft, ob $\varphi(a_1, \ldots, a_n)$ in $\mathcal{M}$ gilt, akzeptiert Sat(Qf Fml ohne Par in $\mathcal{M}$) in polynomieller Zeit.

Um die NP$_\mathcal{M}$-Vollständigkeit bezüglich P$_\mathcal{M}$-Reduktion zu zeigen, zeigen wir

$$\text{Sat(Qf Fml mit Par in } \mathcal{M}) \leq^{\text{P}_\mathcal{M}} \text{Sat(Qf Fml ohne Par in } \mathcal{M}).$$

Die Idee hierbei ist, daß wir für eine Instanz $(\ulcorner\varphi(\bar{x},\bar{y})\urcorner,\bar{a})$ von SAT(QF FML MIT PAR IN $\mathcal{M}$) die Parameter $\bar{a}$ durch den quantorenfreien Typ von $\bar{a}$ ersetzen. Es gilt nämlich für alle $\bar{a}' \in M^{|\bar{a}|}$:

$$\text{Wenn qf-tp}_{\mathcal{M}}[\bar{a}] = \text{qf-tp}_{\mathcal{M}}[\bar{a}'], \text{ so } \mathcal{M} \models \exists\bar{y}\varphi(\bar{a},\bar{y}) \Leftrightarrow \mathcal{M} \models \exists\bar{y}\varphi(\bar{a}',\bar{y}),$$

denn da $\mathcal{M}$ Quantorenelimination erlaubt, gibt es eine zu $\exists\bar{y}\varphi(\bar{x},\bar{y})$ äquivalente quantorenfreie Formel $\psi(x_1,\ldots,x_{|\bar{x}|})$. Weiter gilt dann mit dem vorherigen Lemma $\mathcal{M} \models \exists\bar{y}\varphi(\bar{a},\bar{y})$ genau dann, wenn die Formel ψ aus $\text{qf-tp}_{\mathcal{M}}[\bar{a}]$ beweisbar ist. Dies ist aber mit $\text{qf-tp}_{\mathcal{M}}[\bar{a}] = \text{qf-tp}_{\mathcal{M}}[\bar{a}']$ genau dann der Fall, wenn ψ aus $\text{qf-tp}_{\mathcal{M}}[\bar{a}']$ beweisbar ist, also genau dann, wenn $\mathcal{M} \models \exists\bar{y}\varphi(\bar{a}',\bar{y})$ gilt.

Somit gilt also für ein quantorenfreies $\varphi(\bar{x},\bar{y})$ und $\bar{a} \in M^{|\bar{x}|}$:

$$\mathcal{M} \models \exists\bar{y}\varphi(\bar{a},\bar{y}) \Leftrightarrow \mathcal{M} \models \exists\bar{x}\exists\bar{y}(\text{qf-tp}_{\mathcal{M}}[\bar{a}](\bar{x}) \wedge \varphi(\bar{x},\bar{y})).$$

Die Kodierung der Formel $(\text{qf-tp}_{\mathcal{M}}[\bar{a}](\bar{x}) \wedge \varphi(\bar{x},\bar{y}))$ ist nun eine Instanz des Problems SAT(QF FML OHNE PAR IN $\mathcal{M}$). Zudem ist die Funktion $(\ulcorner\varphi(\bar{x},\bar{y})\urcorner,\bar{a}) \mapsto (\text{qf-tp}_{\mathcal{M}}[\bar{a}] \wedge \varphi(\bar{x},\bar{y}))$ in polynomieller Zeit im Sinne von $\mathcal{M}$ berechenbar, da nach dem vorherigen Lemma 5.1.2 die Funktion $\bar{a} \mapsto \text{qf-tp}_{\mathcal{M}}[\bar{a}]$ in polynomieller Zeit im Sinne von $\mathcal{M}$ berechenbar ist. □

In Strukturen mit endlicher relationaler Symbolmenge gilt also $\mathrm{P}_{\mathcal{M}} \restriction \{0,1\} = \mathrm{P}$ und es gibt ein boolesches $\mathrm{NP}_{\mathcal{M}}$-vollständiges Problem. Die letzte Eigenschaft, die sie erfüllen müssen, um vom H_1-Standardtyp zu sein, ist die Gleichheit von $\mathrm{NP}_{\mathcal{M}}$ und $\mathrm{BNP}_{\mathcal{M}}$. Hierfür gibt es ein schönes Kriterium.

Satz 5.1.4 (stdtp - qftp in np) *Sei $\mathcal{M}$ eine Struktur mit endlicher relationaler Symbolmenge, welche Quantorenelimination erlaubt. Dann ist die Struktur $\mathcal{M}$ genau dann vom H_1-Standardtyp, wenn die Menge der Kodierungen ihrer realisierten quantorenfreien Typen in* NP *liegt.*

Beweis: „→“: Sei $\mathcal{M}$ vom H_1-Standardtyp und sei $\theta := \{\ulcorner\text{qf-tp}_{\mathcal{M}}[\bar{a}]\urcorner \mid \bar{a} \in M^*\}$. Es gilt $\theta \in \mathrm{NP}_{\mathcal{M}} \restriction \{0,1\}$, denn eine nichtdeterministische Maschine im Sinne von $\mathcal{M}$, welche bei Eingabe von $\ulcorner\psi(\bar{x})\urcorner$ testet, ob ψ die richtige Form hat, ein Element $\bar{a} \in M^{|\bar{x}|}$ rät und testet, ob $\mathcal{M} \models \psi(\bar{a})$ gilt, akzeptiert θ in polynomieller Zeit. Da andererseits $\mathcal{M}$ vom H_1-Standardtyp ist, gilt

$$\mathrm{NP}_{\mathcal{M}} \restriction \{0,1\} = \mathrm{BNP}_{\mathcal{M}} \restriction \{0,1\} = \mathrm{NP}$$

und somit $\theta \in \mathrm{NP}$.

„←“: Sei also $\theta := \{\ulcorner\text{qf-tp}_{\mathcal{M}}[\bar{a}]\urcorner \mid \bar{a} \in M^*\}$ und gelte $\theta \in \mathrm{NP}$. Nach Lemma 5.0.10 gilt $\mathrm{P}_{\mathcal{M}} \restriction \{0,1\} = \mathrm{P}$ und nach Lemma 5.1.3 gibt es ein boolesches $\mathrm{NP}_{\mathcal{M}}$-vollständiges Problem. Wir müssen also noch die Gleichheit $\mathrm{NP}_{\mathcal{M}} = \mathrm{BNP}_{\mathcal{M}}$ zeigen. Da das Problem SAT(QF FML OHNE PAR IN $\mathcal{M}$) nach Lemma 5.1.3 bezüglich $\mathrm{P}_{\mathcal{M}}$-Reduktion $\mathrm{NP}_{\mathcal{M}}$-vollständig ist, reicht es hierfür zu zeigen, daß gilt:

$$\text{SAT(QF FML OHNE PAR IN } \mathcal{M}) \in \mathrm{BNP}_{\mathcal{M}}.$$

Ein $\mathrm{BNP}_{\mathcal{M}}$-Algorithmus $\mathbb{A}$, der das Problem SAT(QF FML OHNE PAR IN $\mathcal{M}$) in polynomieller Zeit akzeptiert, arbeitet wie folgt: Bei Eingabe einer Kodierung $\ulcorner\varphi\urcorner$ einer quantorenfreien Formel $\varphi(x_1,\ldots,x_n)$ rät $\mathbb{A}$ eine Kodierung einer quantorenfreien Formel $\psi(x_1,\ldots,x_n)$ der Länge $\max\{|\ulcorner r(x_1,\ldots,x_n)\urcorner| \mid r \in \theta\}$ und prüft mit Hilfe des NP-Algorithmus für θ nach, ob ψ ein in $\mathcal{M}$ realisierter quantorenfreier Typ ist. Ist dies nicht der Fall, so verwirft $\mathbb{A}$ die Eingabe. Andernfalls prüft der Algorithmus, ob die Formel φ aus ψ beweisbar ist. Hierzu reicht es, da φ eine quantorenfreie Formel und somit eine boolesche Kombination von atomaren Formeln ist, wenn $\mathbb{A}$ die Wahrheitswerte der in φ vorkommenden atomaren oder negiert atomaren Subformeln aus ψ gewinnt und dann mit diesen und mit Hilfe der mittels der Selektorfunktion simulierten booleschen Funktionen den Wahrheitswert von φ errechnet. Wenn der so errechnete Wert 1 ist, ist φ aus ψ beweisbar, und für dasjenige $\bar{a} \in M^*$ mit $\psi = \text{qf-tp}_{\mathcal{M}}[\bar{a}]$ gilt mit Lemma 5.1.2 somit $\mathcal{M} \models \varphi(\bar{a})$. In diesem Fall akzeptiert $\mathbb{A}$ die Eingabe. Ist der errechnete Wert 0, so verwirft $\mathbb{A}$ die Eingabe.

Da der Algorithmus insgesamt eine 0-1-Folge polynomieller Länge rät und mit der verwendeten Methode in polynomieller Zeit nachprüfen kann, ob die eingegebene Formel von einem den geratenen quantorenfreien Typ erfüllenden Element erfüllt wird, ist somit SAT(QF FML OHNE PAR IN $\mathcal{M}$) $\in \mathrm{BNP}_{\mathcal{M}}$ und damit $\mathrm{NP}_{\mathcal{M}} = \mathrm{BNP}_{\mathcal{M}}$. Die Struktur $\mathcal{M}$ ist also vom H_1-Standardtyp. □

5.2 Lineare Wörterbücher

Im folgenden werden wir die Struktur der linearen Wörterbücher LW betrachten. Wir werden sehen, daß LW vom H_1-Standardtyp ist. Leider ist nicht bekannt, ob $\mathrm{P}_{\mathrm{LW}} = \mathrm{NP}_{\mathrm{LW}}$ gilt.

Das Universum von LW ist die Menge

$$U := \{0,1\} \cup \{(\varepsilon_n)_{n\in\mathbb{N}} \mid \varepsilon_i \in \{0,1\} \text{ für alle } i \text{ und } (\varepsilon_n)_n \text{ schließlich aperiodisch}\}$$

der unendlichen schließlich aperiodischen 0-1-Folgen mit 0 und 1. Die Struktur LW besitzt (außer der Selektorfunktion und der Identitätsfunktion) die Funktion v, welche einer unendlichen 0-1-Folge ihr erstes Folgenglied zuordnet und definiert ist durch

$$v(x) := \begin{cases} x, & \text{falls } x \in \{0,1\} \\ \varepsilon_0, & \text{falls } x = (\varepsilon_n)_{n\in\mathbb{N}} \end{cases}$$

Außerdem besitzt LW die Funktion h, welche einer unendlichen 0-1-Folge dieselbe Folge ohne ihr erstes Folgeglied zuordnet:

$$h(x) := \begin{cases} x, & \text{falls } x \in \{0,1\} \\ (\varepsilon_{n+1})_{n\in\mathbb{N}}, & \text{falls } x = (\varepsilon_n)_{n\in\mathbb{N}} \end{cases}$$

Da die beiden Funktionen v und h auf der Menge $\{0,1\}$ die Identitätsfunktion berechnen, erhalten wir mit den zu Beginn dieses Kapitels angestellten Überlegungen sofort das folgende Lemma.

Lemma 5.2.1 *Für die Struktur* LW *gilt* $\mathrm{P}_{\mathrm{LW}} \restriction \{0,1\} = \mathrm{P}$.

Beweis: Da die Funktionen $v \restriction \{0,1\}$ und $h \restriction \{0,1\}$ gleich der Identität sind, ist die von $\{0,1\}$ in LW erzeugte Substruktur endlich. Mit Lemma 5.0.10 gilt dann $\mathrm{P}_{\mathrm{LW}} \restriction \{0,1\} = \mathrm{P}$. □

Bemerkung 5.2.2 Es läßt sich zeigen, daß für die mit Konstantensymbolen für jedes Element des Universums U verlängerte Struktur $(\mathrm{LW}, (c_u)_{u\in U})$ die Gleichheit

$$\mathrm{P}_{(\mathrm{LW},(c_u)_{u\in U})} \restriction \{0,1\} = \mathbb{P}$$

gilt (siehe [22]). Da aber $\mathrm{P} \neq \mathbb{P}$ gilt, gilt somit auch

$$\mathrm{P}_{\mathrm{LW}} \neq \mathrm{P}_{(\mathrm{LW},(c_u)_{u\in U})}.$$

Berechnungen im Sinne von LW unterscheiden sich somit stark von Berechnungen im Sinne von $(\mathrm{LW}, (c_u)_{u\in U})$.

Um zu zeigen, daß LW vom H_1-Standardtyp ist, müssen wir also noch die Gleichheit $\mathrm{NP}_{\mathrm{LW}} = \mathrm{BNP}_{\mathrm{LW}}$ und die Existenz eines booleschen bezüglich P_{LW}-Reduktion $\mathrm{NP}_{\mathrm{LW}}$-vollständigen Problems zeigen. Hierzu gehen wir ähnlich vor wie bei den Beweisen zu Lemma 5.1.3 und Satz 5.1.4. Wir zeigen, daß es um zu sehen, ob eine quantorenfreie Formel mit Parametern erfüllbar ist, reicht, einen bestimmten Typ statt eines konkreten Wortes zu raten. Dieser Typ kann dann wieder binär kodiert werden und somit erhalten wir die beiden gesuchten Eigenschaften von LW.

Zunächst definieren wir den n-Typ eines Wortes aus U^*:

Definition 5.2.3 (n-Typ) Für $n \in \mathbb{N}$ ist der *n-Typ* $n\text{-tp}[\bar{a}]$ eines Wortes $\bar{a}$ aus U^* eine Konjunktion über die Gleichungen

- $v(h^m(x_i)) = \left\{ \begin{array}{ll} 0, & \text{falls } v(h^m(a_i)) = 0 \\ 1, & \text{falls } v(h^m(a_i)) = 1 \end{array} \right\}$ für $m \leq n$ und $i \in [|\bar{a}|]$
- $\left. \begin{array}{ll} h^{m_1}(x_i) = h^{m_2}(x_j), & \text{falls } h^{m_1}(a_i) = h^{m_2}(a_j) \\ h^{m_1}(x_i) \neq h^{m_2}(x_j), & \text{falls } h^{m_1}(a_i) \neq h^{m_2}(a_j) \end{array} \right\}$ für $m_1, m_2 \leq n$ und $i, j \in [|\bar{a}|]$.

Der n-Typ von $\bar{a}$ bestimmt also eindeutig die ersten n Folgeglieder der Komponenten von $\bar{a}$ und ob zwei der Komponenten von $\bar{a}$ ab dem i-ten und j-ten Folgenglied identisch sind, wobei i und j höchstens so groß wie n sind. Analog zum früher definierten quantorenfreien Typ hat der n-Typ dann gewisse nützliche Eigenschaften.

Lemma 5.2.4 (Eigenschaften des n-Typs) *Sei n eine natürliche Zahl. Dann gilt:*

(i) die Funktion $\bar{y} \mapsto n\text{-tp}[\bar{y}]$ ist in polynomieller Zeit in $n + |\bar{y}|$ berechenbar

(ii) es ist in polynomieller Zeit in $|\varphi| + n$ entscheidbar, ob eine Formel φ ein n-Typ ist

(iii) für quantorenfreies $\varphi(\bar{x})$ und $\bar{a} \in U^{|\bar{x}|}$ gilt $\mathrm{LW} \models \varphi(\bar{a})$ *genau dann, wenn φ aus $|\varphi|\text{-tp}[\bar{a}]$ beweisbar ist.*

Beweis: *Zu (i):* Um den n-Typ eines Wortes $\bar{y}$ zu berechnen, werden zunächst in $\mathcal{O}(n \cdot |\bar{y}|)$ Schritten für jede Komponente y_i von $\bar{y}$ die Werte $h^0(y_i), \ldots, h^n(y_i)$ berechnet. Dann wird für jedes $h^l(y_i)$ der Wert $v(h^l(y_i))$ berechnet und je nach Ergebnis die Formel $v(h^l(x_i)) = 0$ oder $v(h^l(x_i)) = 1$ zu der Konjunktion hinzugefügt. Abschließend werden in $\mathcal{O}(n^2 \cdot |\bar{y}|^2)$ Schritten die Gleichheiten der Form $h^{m_1}(y_i) = h^{m_2}(y_j)$ überprüft und je nach Ergebnis die Formel $h^{m_1}(x_i) = h^{m_2}(x_j)$ oder die Formel $h^{m_1}(x_i) \neq h^{m_2}(x_j)$ zu der Konjunktion hinzugefügt.

Zu (ii): Um bei Eingabe einer Formel φ zu entscheiden, ob $\varphi = n\text{-tp}[\bar{a}]$ für ein $\bar{a} \in U^*$ gilt, reicht es, zu überprüfen, ob die Formel φ die richtige Form hat, ob zwischen den Gleichungen keine Widersprüche auftreten und ob keine Gleichungen der Form $h^{m_1}(x_i) = h^{m_2}(x_i)$ auftreten, welche x_i periodisch machen würden. Dies ist in polynomieller Zeit möglich.

Zu (iii): Wenn φ quantorenfrei ist, so kommen in φ also auch höchstens Subformeln der Form $v(h^{s_i}(x_i) = \varepsilon$ oder $h^{s_i}(x_i) = h^{s_j}(x_j)$ mit $s_i, s_j \leq |\varphi|$ vor (bis auf Subformeln mit der Identitätsfunktion beziehungsweise der Selektorfunktion, welche sich aber in derartige Subformeln umwandeln lassen). Ob diese Gleichungen von $\bar{a}$ erfüllt werden oder nicht, ist durch den $|\varphi|$-Typ von $\bar{a}$ festgelegt. □

Um das nächste, für das gesuchte Ergebnis grundlegende Lemma zu beweisen, benötigen wir noch eine technische Definition.

Definition 5.2.5 (n-abhängig) Sei n eine natürliche Zahl. Zwei Elemente a_1, a_2 aus U heißen *n-abhängig*, falls es natürliche Zahlen m_1, m_2 mit $m_1 \leq n$ und $m_2 \leq n$ gibt, so daß $h^{m_1}(a_1) = h^{m_2}(a_2)$ gilt.

Das folgende Lemma wird uns erlauben, das Problem SAT(QF FML MIT PAR IN LW) auf das Problem SAT(QF FML OHNE PAR IN LW) zurückzuführen. Mit der *Länge einer Formel* ist hierbei die Anzahl der Zeichen der Formel gemeint.

Lemma 5.2.6 *Sei n eine natürliche Zahl. Wenn zwei Elemente $\bar{a_1}$ und $\bar{a_2}$ von U^* denselben $(n^2 + n)$-Typ haben, so erfüllen sie dieselben existentiellen Formeln der Länge n: für alle existentiellen Formeln $\exists \bar{y} \varphi(\bar{x}, \bar{y})$ der Länge n im Sinne von* LW *gilt*

$$\text{LW} \models \exists \bar{y} \varphi(\bar{a_1}, \bar{y}) \Leftrightarrow \text{LW} \models \exists \bar{y} \varphi(\bar{a_2}, \bar{y}).$$

Beweis: Betrachten wir zunächst für eine natürliche Zahl k eine quantorenfreie Formel der Länge k. Da die Formel die Länge k hat, kommen in ihr auch höchstens k-fach verschachtelte Terme vor. Zudem lassen sich die Selektorfunktion und die Identitätsfunktion durch die beiden Funktionen v und h ausdrücken, hierbei kommen v und h nicht verschachtelt vor. Die für die Frage, ob ein Element aus U^* die quantorenfreie Formel erfüllt oder nicht, relevanten Informationen sind somit alle im k-Typ dieses Elements enthalten. Wenn also zwei Elemente denselben k-Typ haben, so erfüllen sie auch dieselben quantorenfreien Formeln der Länge höchstens k.

Bei einer existentiellen Formel $\exists \bar{y} \varphi(\bar{x}, \bar{y})$ der Länge n werden andererseits höchstens n Variablen quantifiziert, und der quantorenfreie Teil φ hat höchstens die Länge n. Um die Behauptung zu zeigen, reicht es, die Behauptung

$$\text{für alle } \bar{b_1} \in U^n \text{ gibt es ein } \bar{b_2} \in U^n \text{ mit } n\text{-tp}[\bar{a_1}, \bar{b_1}] = n\text{-tp}[\bar{a_2}, \bar{b_2}]$$

zu zeigen. Denn wenn für $\bar{a_1}$ dann LW $\models \exists \bar{y} \varphi(\bar{a_1}, \bar{y})$ gilt, so gibt es ein $\bar{b_1} \in U^n$ mit LW $\models \varphi(\bar{a_1}, \bar{b_1})$. Da es damit ein $\bar{b_2} \in U^*$ gibt, so daß n-tp$[\bar{a_1}, \bar{b_1}] = n$-tp$[\bar{a_2}, \bar{b_2}]$ gilt und da nach obiger Überlegung zwei Worte desselben n-Typs dieselben quantorenfreien Formeln der Länge n erfüllen, gilt damit auch LW $\models \varphi(\bar{a_2}, \bar{b_2})$ und somit auch LW $\models \exists \bar{y} \varphi(\bar{a_2}, \bar{y})$.

Um ein solches Wort $\bar{b_2}$ zu konstruieren, gehen wir folgendermaßen vor:
Seien $\bar{a_1}$ und $\bar{a_2}$ mit (n^2+n)-tp$[\bar{a_1}] = (n^2+n)$-tp$[\bar{a_2}]$ und $\bar{b_1} \in U^n$ gegeben. Um die Komponenten von $\bar{b_1}$, welche eine gewisse relevante Verbindung zu $\bar{a_1}$ haben, zu kennzeichnen, nennen wir ein Element b_{1i_1} von $\bar{b_1}$ dann *n-zugänglich*, wenn es (nicht notwendigerweise verschiedene) Elemente $b_{1i_2}, \ldots, b_{1i_n}$ von $\bar{b_1}$ und ein a_{1i_0} aus $\bar{a_1}$ gibt, so daß die beiden Elemente b_{1i_n} und a_{1i_0} und für $1 \leq j \leq n-1$ jeweils die Elemente b_{1i_j} und $b_{1i_{j+1}}$ im Sinne der vorhergehenden Definition n-abhängig sind.

Wir behandeln zunächst die n-zugänglichen Komponenten von $\bar{b_1}$. Ohne Einschränkung der Allgemeinheit seien $b_{11}, \ldots, b_{1k}$ die n-zugänglichen Komponenten von $\bar{b_1}$. Für ein solches n-zugängliches b_{1i} gibt es eine minimale natürliche Zahl m und eine natürliche Zahl l mit $m \leq n^2$ und $l \leq n^2$, sowie ein $\bar{\varepsilon_i} \in \{0,1\}^m$ und ein a_{1i_0} aus $\bar{a_1}$, so daß $b_{1i} = \bar{\varepsilon_i} h^l(a_{1i_0})$ gilt. Um die entsprechende Komponente b_{2i} zu konstruieren setzen wir

$$b_{2i} := \bar{\varepsilon_i} h^l(a_{2i_0}).$$

Dann ist n-tp$[\bar{a_1}, b_{11}, \ldots, b_{1k}] = n$-tp$[\bar{a_2}, b_{21}, \ldots, b_{2k}]$, denn:

- Falls eine Gleichung der Form $v(h^s(x_i)) = \varepsilon$ in n-tp$[\bar{a_1}, b_{11}, \ldots, b_{1k}]$ vorkommt, so ist entweder $s \leq |\bar{a_1}|$ und die Gleichung kommt in (n^2+n)-tp$[\bar{a_1}]$ und somit auch in (n^2+n)-tp$[\bar{a_2}]$ vor, weswegen $v(h^s(a_{2i})) = \varepsilon$ gilt und die Gleichung auch in n-tp$[\bar{a_2}, b_{21}, \ldots, b_{2k}]$ vorkommt; oder es ist $i > |\bar{a_1}|$. Dann gilt allerdings für das entsprechende b_{1j} die Gleichheit $v(h^s(b_{1j})) = \varepsilon$ und aufgrund der n-Zugänglichkeit von b_{1j} gilt somit für geeignete $m, l \in \mathbb{N}$ mit $m \leq n^2$ und $l \leq n$, ein $\bar{\varepsilon_j} \in \{0,1\}^m$ und ein a_{1j_0} aus $\bar{a_1}$ die Gleichheit $v(h^s(\bar{\varepsilon_j} h^l(a_{1j_0}))) = \varepsilon$. Nun gibt es zwei Fälle:
  - Erster Fall: $s < m$. Da $|\bar{\varepsilon_j}| = m$ gilt, gilt mit $v(h^s(\bar{\varepsilon_j} h^l(a_{1j_0}))) = \varepsilon$ die Gleichung $\varepsilon_{js+1} = \varepsilon$ und somit auch $v(h^s(\bar{\varepsilon_j} h^l(a_{2j_0}))) = \varepsilon$. Nach Konstruktion von b_{2j} gilt somit $v(h^s(b_{2j})) = \varepsilon$ und die Gleichung $v(h^s(x_i)) = \varepsilon$ kommt in n-tp$[\bar{a_2}, b_{21}, \ldots, b_{2k}]$ vor.
  - Zweiter Fall: $s \geq m$. Mit $s \geq m$ und $v(h^s(\bar{\varepsilon_j} h^l(a_{1j_0}))) = \varepsilon$ gilt dann auch $v(h^{l+s-m}(a_{1j_0})) = \varepsilon$. Da $l + s - m \leq n^2 + n$ und die (n^2+n)-Typen von $\bar{a_1}$ und $\bar{a_2}$ identisch sind gilt somit auch $v(h^{l+s-m}(a_{2j_0})) = \varepsilon$ und damit auch $v(h^s(b_{2j})) = \varepsilon$. Die Gleichung kommt also auch in n-tp$[\bar{a_2}, b_{21}, \ldots, b_{2k}]$ vor.
- Falls für $s, r \leq n$ eine Gleichung oder Ungleichung der Form $h^s(x_i) = h^r(x_j)$ in n-tp$[\bar{a_1}, b_{11}, \ldots, b_{1k}]$ vorkommt, so gibt es wieder verschiedene Fälle. Wir beschränken uns hier auf den Fall, daß die jeweilige Gleichung vorkommt, die Beweise für die Ungleichungen laufen analog.
  - Erster Fall: $i, j \leq |\bar{a_1}|$. In diesem Fall macht die Gleichung nur eine Aussage über eine Beziehung zwischen a_{1i} und a_{1j}. Es gilt also $h^s(a_{1i}) = h^r(a_{1j})$. Somit kommt die Gleichung auch in (n^2+n)-tp$[\bar{a_1}]$ und also auch in (n^2+n)-tp$[\bar{a_2}]$ vor. Damit gilt aber auch $h^s(a_{2i}) = h^r(a_{2j})$ und die Gleichung kommt auch in n-tp$[\bar{a_2}, b_{21}, \ldots, b_{2k}]$ vor.

- Zweiter Fall: $i \leq |\bar{a_1}|$ und $j > |\bar{a_1}|$ oder $j \leq |\bar{a_1}|$ und $i > |\bar{a_1}|$. Ohne Einschränkung sei $i \leq |\bar{a_1}|$ und $j > |\bar{a_1}|$. Es gilt also $h^s(b_{1t}) = h^r(a_{1j})$ für das entsprechende b_{1t}. Für geeignete $m, l, \bar{\varepsilon_t}, a_{1j_0}$ ist wieder $b_{1t} = \bar{\varepsilon_t} h^l(a_{1j_0})$.
  - Falls $s < m$, so ist $h^s(b_{1t}) = \varepsilon_{t(s+1)} \dots \varepsilon_{tm} h^l(a_{1j_0})$ und somit $h^l(a_{1j_0}) = h^{r+m-s}(a_{1j})$. Da $l \leq n$ und $r + m - s \leq n^2$ gilt, kommt somit $h^l(x_{j_0}) = h^{r+m-s}(x_j)$ in (n^2+n)-tp$[\bar{a_1}]$ und also auch in (n^2+n)-tp$[\bar{a_2}]$ vor. Es gilt also $h^l(a_{2j_0}) = h^{r+m-s}(a_{2j})$. Andererseits gilt aber auch $v(h^r(a_{1j})) = \varepsilon_{t(s+1)}$, $v(h^{r+1}(a_{1j})) = \varepsilon_{t(s+2)}, \dots, v(h^{r+m-s-1}(a_{1j})) = \varepsilon_{tm}$, die entsprechenden Gleichungen kommen also auch in $(n^2 + n)$-tp$[\bar{a_1}]$ und somit in $(n^2 + n)$-tp$[\bar{a_2}]$ vor, und es gilt $v(h^r(a_{2j})) = \varepsilon_{t(s+1)}$, $v(h^{r+1}(a_{2j})) = \varepsilon_{t(s+2)}, \dots,$ $v(h^{r+m-s-1}(a_{2j})) = \varepsilon_{tm}$. Insgesamt gilt also $h^s(b_{1t}) = h^r(a_{2j})$ und die zugehörige Gleichung kommt in n-tp$[\bar{a_2}, b_{21}, \dots, b_{2k}]$ vor.
  - Falls $s \geq m$, so ist $h^s(b_{1t}) = h^{l+s-m}(a_{1j_0})$. Eine entsprechende Gleichung kommt somit auch in $(n^2 + n)$-tp$[\bar{a_1}] = (n^2 + n)$-tp$[\bar{a_2}]$ vor und es gilt $h^s(b_{2t}) = h^{l+s-m}(a_{2j_0})$. Die Gleichung kommt somit in n-tp$[\bar{a_2}, b_{21}, \dots, b_{2k}]$ vor.
- Dritter Fall: $i, j > |\bar{a_1}|$. Es gilt also $h^s(b_{1i'}) = h^r(b_{1j'})$. Da $b_{1i'}$ und $b_{1j'}$ beide n-zugänglich sind, ist $b_{1i'} = \bar{\varepsilon_{i'}} h^{l_{i'}}(a_{1i'_0})$ und $b_{1j'} = \bar{\varepsilon_{j'}} h^{l_{j'}}(a_{1j'_0})$. Analog zum zweiten Fall kommt dann die Gleichung $h^s(x_i) = h^r(x_j)$ auch in n-tp$[\bar{a_2}, b_{21}, \dots, b_{2k}]$ vor.

Betrachten wir nun die nicht n-zugänglichen Komponenten $b_{1(k+1)} \dots b_{1n}$ von $\bar{b_1}$. Da diese nicht n-zugänglich sind, sind sie auch nicht n-abhängig von $\bar{a_1} b_{11} \dots b_{1k}$, das heißt in n-tp$[\bar{a_1}\bar{b_1}]$ kommen für alle $i \in [|\bar{a_1}| + k]$ und $j \in \{|\bar{a_1}| + k + 1, \dots, |\bar{a_1}| + n\}$ und für alle $s_i, s_j \in [n]$ die Ungleichungen $h^{s_i}(b_{1i}) \neq h^{s_j}(b_{1j})$ vor. Wir müssen also Elemente $b_{2(k+1)}, \dots, b_{2n}$ konstruieren, die n-unabhängig von $\bar{a_2} b_{21} \dots b_{2k}$ sind, und die dieselben Gleichungen $v(h^s(x_i)) = \varepsilon$ und $h^s(x_i) = h^r(x_j)$ erfüllen wie die $b_{1(k+1)}, \dots, b_{1n}$.

Die ersten n Folgeglieder dieser Elemente sind dabei schon durch die Gleichungen der Form $v(h^s(x_i))$ festgelegt. Die Idee ist die, daß wir danach eine endliche 0-1-Folge einfügen, die die n-Unabhängigkeit jedes der $b_{2(k+1)}, \dots, b_{2n}$ von $\bar{a_2} b_{21} \dots b_{2k}$ sicherstellt. Danach füllen wir (unter Beachtung der Gleichungen der Form $h^s(x_i) = h^r(x_j)$) mit beliebigen 0-1-Folgen auf. Hierzu beginnen wir mit demjenigen b_{1i_1} welches in den Gleichungen $h^s(x_i) = h^r(x_j)$ „den kleinsten Exponenten hat“, für welches also für alle b_{1j} und alle s_i, s_j mit $h^{s_i}(b_{1i_1}) = h^{s_j}(b_{1j})$ gilt: $s_i \leq s_j$. Da die b_{1i} schließlich aperiodisch sind, existiert solch ein Element b_{1i_1}. Falls es mehrere solche Elemente geben sollte, wählen wir ein beliebiges. Die ersten n Folgenglieder des zu konstruierenden b_{2i_1} werden durch die von b_{1i_1} erfüllten Gleichungen der Form $v(h^s(x_i))$ festgelegt. Danach fügen wir eine endliche 0-1-Folge ein, welche die n^2-Unabhängigkeit des b_{2i_1} von $\bar{a_2} b_{21} \dots b_{2k}$ garantiert. Da wir insgesamt höchstens n Elemente konstruieren müssen, wird damit sichergestellt, daß auch das letzte konstruierte Element n-unabhängig von $\bar{a_2} b_{21} \dots b_{2k}$ ist.

Als nächstes betrachten wir das Element b_{1i_2}, welches in den Gleichungen $h^s(x_i) = h^r(x_j)$ „den nächstkleineren Exponenten hat“. Für die Konstruktion des Elementes b_{2i_2} halten wir uns für die ersten n Folgenglieder wieder zunächst an die Gleichungen der Form $v(h^s(x_i))$, welche von b_{1i_2} erfüllt werden. Den Rest von b_{2i_2} konstruieren wir aus dem Element b_{2i_1} mit Hilfe der entsprechenden Gleichung $h^{s_i}(x_i) = h^{s_j}(x_j)$. Sollte b_{1i_2} nicht

mit b_{1i_1} über n-abhängige b_j zugänglich sein, so verfahren wir wie bei der Konstruktion von b_{2i_1}. Analog konstruieren wir die übrigen Elemente von $b_{2(k+1)} \dots b_{2n}$.

Die so konstruierten Elemente $b_{2(k+1)} \dots b_{2n}$ erfüllen dann dieselben Gleichungen der Form $h^s(x_i) = h^r(x_j)$ wie $b_{1(k+1)} \dots b_{1n}$ und sind n-unabhängig von $\bar{a_2} b_{21} \dots b_{2k}$. Somit gilt insgesamt

$$n\text{-tp}[\bar{a_1}, \bar{b_1}] = n\text{-tp}[\bar{a_2}, \bar{b_2}].$$

Es gibt also für alle $\bar{b_1} \in U^n$ ein $\bar{b_2} \in U^n$ mit n-tp$[\bar{a_1}, \bar{b_1}] = n$-tp$[\bar{a_2}, \bar{b_2}]$. Mit den zu Beginn des Beweises angestellten Überlegungen folgt hieraus die Behauptung. □

Nun können wir endlich das erwünschte Ergebnis beweisen.

Satz 5.2.7 *Die Struktur* LW *ist vom H_1-Standardtyp.*

Beweis: Zunächst gilt in LW nach Lemma 5.2.1 die Gleichheit $\mathrm{P}_{\mathrm{LW}} \restriction \{0,1\} = \mathrm{P}$. Für die Existenz eines booleschen bezüglich P_{LW}-Reduktion $\mathrm{NP}_{\mathrm{LW}}$-vollständigen Problems zeigen wir zunächst

$$\textsc{Sat}(\textsc{Qf Fml mit Par in LW}) \leq^{\mathrm{P}_{\mathrm{LW}}} \textsc{Sat}(\textsc{Qf Fml ohne Par in LW}).$$

Da das Problem Sat(Qf Fml mit Par in LW) mit Satz 4.2.9 bezüglich P_{LW}-Reduktion $\mathrm{NP}_{\mathrm{LW}}$-vollständig ist, folgt damit die $\mathrm{NP}_{\mathrm{LW}}$-Vollständigkeit von Sat(Qf Fml ohne Par in LW).

Sei also $\mathbb{A}$ ein Algorithmus, welcher bei Eingabe einer Instanz $(\ulcorner \varphi(\bar{x}, \bar{y}) \urcorner, \bar{a})$ von Sat(Qf Fml mit Par in LW) zunächst den Wert $|\varphi|$ und dann die Formel $(|\varphi|^2 + |\varphi|)$-tp$[\bar{a}](\bar{x})$ berechnet. Dies gelingt mit Lemma 5.2.4 in polynomieller Zeit. Anschließend bildet $\mathbb{A}$ wiederum in polynomieller Zeit die Kodierung der Formel $((|\varphi|^2 + |\varphi|)\text{-tp}[\bar{a}](\bar{x}) \wedge \varphi(\bar{x}, \bar{y}))$. Dies ist nun eine Instanz von Sat(Qf Fml ohne Par in LW), und da mit Lemma 5.2.6 zwei Tupel mit demselben (n^2+n)-Typ dieselben existentiellen Formeln der Länge n erfüllen, gilt

$$\mathrm{LW} \models \exists \bar{y} \varphi(\bar{a}, \bar{y}) \Leftrightarrow \mathrm{LW} \models \exists \bar{x} \exists \bar{y}((|\varphi|^2 + |\varphi|)\text{-tp}[\bar{a}](\bar{x}) \wedge \varphi(\bar{x}, \bar{y})).$$

Der Algorithmus $\mathbb{A}$ ist somit eine P_{LW}-Reduktion von Sat(Qf Fml mit Par in LW) auf Sat(Qf Fml ohne Par in LW). Desweiteren sind quantorenfreie Formeln ohne Parameter binär kodierbar, da die Symbolmenge von LW endlich ist. Das Problem Sat(Qf Fml ohne Par in LW) ist somit ein bezüglich P_{LW}-Reduktion $\mathrm{NP}_{\mathrm{LW}}$-vollständiges boolesches Problem.

Die Gleichheit von $\mathrm{NP}_{\mathrm{LW}}$ und $\mathrm{BNP}_{\mathrm{LW}}$ gilt andererseits, da ein Algorithmus, welcher bei Eingabe einer Kodierung einer quantorenfreien Formel φ die binäre Kodierung eines $|\varphi|$-Typs rät und dann mittels Wahrheitswertrechnens in polynomieller Zeit überprüft, ob φ aus dem geratenen Typ beweisbar ist, mit Lemma 5.2.4 gerade das Problem Sat(Qf Fml ohne Par in LW) akzeptiert. Da der Algorithmus in polynomieller Zeit läuft, gilt somit Sat(Qf Fml ohne Par in LW) $\in \mathrm{BNP}_{\mathrm{LW}}$ und mit den obigen Überlegungen $\mathrm{NP}_{\mathrm{LW}} = \mathrm{BNP}_{\mathrm{LW}}$. □

Mit Korollar 4.3.9 ist somit die Frage, ob $\mathrm{P}_{\mathrm{LW}} = \mathrm{NP}_{\mathrm{LW}}$ gilt, äquivalent zu der klassischen P = NP-Frage. Ebenso ist die Frage, ob $\mathrm{NP}_{\mathrm{LW}} = co\mathrm{NP}_{\mathrm{LW}}$ gilt, äquivalent zu der Frage, ob NP = *co*NP gilt. Leider ist für die linearen Wörterbücher keine der Antworten auf diese Fragen bekannt.

Bemerkung 5.2.8 Es läßt sich zeigen, daß die Struktur $(\mathrm{LW}, (c_u)_{u\in U})$ vom $\mathbb{H}_1$-Standardtyp ist. Die Frage, ob $\mathrm{P}_{(\mathrm{LW},(c_u)_{u\in U})} = \mathrm{NP}_{(\mathrm{LW},(c_u)_{u\in U})}$ gilt, ist somit äquivalent zu der klassischen Frage, ob $\mathbb{P} = \mathbb{NP}$ gilt. Hierfür siehe auch Bemerkung 5.2.2 und [22].

5.3 Strukturen über den reellen Zahlen

Wir wollen nun verschiedene Strukturen über den reellen Zahlen betrachten. Zunächst zu den reellen Zahlen mit der Addition und dem additiven Inversen.

5.3.1 Mit Addition und additivem Inversen

Wir betrachten die Struktur $\mathcal{R}^{=} = (\mathbb{R}, +, -)$, in deren Symbolmenge (neben dem dreistelligen Funktionssymbol für die Selektorfunktion, dem einstelligen Funktionssymbol für die Identität und den beiden Konstantensymbolen für 0 und 1) nur ein zweistelliges Funktionssymbol „+“ für die Addition und ein einstelliges Funktionssymbol „−“, für die Inversenfunktion, welche einer reellen Zahl ihr additives Inverses zuordnet, vorkommen. Unser Ziel ist es wieder, zu untersuchen, ob die Struktur $\mathcal{R}^{=}$ vom H_1-Standardtyp ist.

In einem ersten Schritt betrachten wir die Klasse $\mathrm{P}_{\mathcal{R}^{=}} \restriction \{0,1\}$. Da eine deterministische Turingmaschine über $\mathcal{R}^{=}$ bei Eingabe von $\bar{x} \in \{0,1\}^*$ in ihrer Berechnung nur ganze Zahlen erzeugen und verwenden kann, ist das folgende Resultat nicht wirklich überraschend.

Satz 5.3.1 *In der Struktur $\mathcal{R}^{=}$ gilt:*

$$\mathrm{P}_{\mathcal{R}^{=}} \restriction \{0,1\} = \mathrm{P}.$$

Beweis: Da die booleschen Funktionen in $\mathcal{R}^{=}$ mit der Selektorfunktion simuliert werden können, gilt die Inklusion $\mathrm{P} \subseteq \mathrm{P}_{\mathcal{R}^{=}}$ sofort, es ist also nur noch die Inklusion $\mathrm{P}_{\mathcal{R}^{=}} \restriction \{0,1\} \subseteq \mathrm{P}$ zu zeigen.

Sei also X ein Problem aus $\mathrm{P}_{\mathcal{R}^{=}} \restriction \{0,1\}$. Dann gibt es ein Problem $X' \in \mathrm{P}_{\mathcal{R}^{=}}$, so daß $X = X' \cap \{0,1\}^*$ gilt. Sei nun $\mathbb{A}$ die deterministische Turingmaschine im Sinne von $\mathcal{R}^{=}$, welche X' in Zeit $p(n)$ entscheidet. Bei Eingabe von $\bar{x} \in \{0,1\}^*$ kommen in der Berechnung von $\mathbb{A}$ nur ganze Zahlen vor. Desweiteren kann $\mathbb{A}$ im ungünstigsten Fall die Zahl $2^{p(n)}$ erzeugen, indem sie $p(n)$-mal die Funktion $x \mapsto 2x$ iteriert. Für alle in der Berechnung vorkommenden Zahlen z gilt also $|z| \leq 2^{p(n)}$. Damit können aber alle in der Berechnung vorkommenden Zahlen in 0-1-Folgen der Länge höchstens $\log(2^{p(n)}) + 1 = p(n) + 1$ kodiert werden.

Eine klassische Turingmaschine, welche die Maschine $\mathbb{A}$ bei Eingabe von $\bar{\varepsilon} \in \{0,1\}^*$ simuliert, indem sie die entsprechenden binären Kodierungen verwendet, benötigt also mit einer geeigneten Konstanten $C \in \mathbb{N}$ für jede Addition und jede Berechnung eines Inversen

jeweils höchstens $C \cdot p(|\bar{\varepsilon}|)$ Schritte. Da die Maschine $\mathbb{A}$ nach höchstens $p(|\bar{\varepsilon}|)$ Schritten hält, hält die Maschine $\mathbb{A}'$ nach höchstens $C \cdot (p(|\bar{\varepsilon}|))^2$ Schritten. Somit gilt $X \in \mathrm{P}$. □

Auch hinsichtlich des zweiten Kriteriums für den H_1-Standardtyp erhalten wir ein positives Ergebnis.

Satz 5.3.2 *In der Struktur $\mathcal{R}^=$ gilt:*

$$\mathrm{NP}_{\mathcal{R}^=} = \mathrm{BNP}_{\mathcal{R}^=}.$$

Beweis: Mit Lemma 4.1.9 ist klar, daß $\mathrm{BNP}_{\mathcal{R}^=} \subseteq \mathrm{NP}_{\mathcal{R}^=}$ gilt. Um die Inklusion $\mathrm{BNP}_{\mathcal{R}^=} \subseteq \mathrm{NP}_{\mathcal{R}^=}$ zu zeigen, reicht es mit Satz 4.2.18 zu zeigen, daß das Problem der Erfüllbarkeit von rudimentär-plus Formeln in der Klasse $\mathrm{BNP}_{\mathcal{R}^=}$ ist.

Das Problem SAT(RUD+ FML MIT PAR IN $\mathcal{R}^=$) ist sogar in der Klasse $\mathrm{P}_{\mathcal{R}^=}$, denn: Eine rudimentär-plus Formel ist eine Konjunktion von Gleichungen und Ungleichungen und läßt sich somit in polynomieller Zeit in ein System von linearen Gleichungen und Ungleichungen umwandeln, welches ganzzahligen Koeffizienten hat und auf dessen rechter Seite reelle Zahlen stehen. Falls in einem der Konjunktionsglieder die Selektorfunktion vorkommt, wird sie dabei wie folgt umgewandelt:

$$\begin{array}{lcl} S(x_1,x_2,x_3) = x_2 & \rightarrow & x_1 = 0 \\ S(x_1,x_2,x_3) = x_3 & \rightarrow & x_1 = 1 \\ S(x_1,x_2,x_3) = x_1 & \rightarrow & \left\{ \begin{array}{l} x_1 \neq 0 \\ x_1 \neq 1 \end{array} \right. \\ S(x_1,x_2,x_3) = x_4 & \rightarrow & x_1 \neq x_1 \text{ für } x_4 \notin \{x_1,x_2,x_3\} \end{array}$$

Das so konstruierte System ist genau dann lösbar, wenn die ursprüngliche rudimentär-plus Formel erfüllbar ist. Indem rationale Zahlen als Tupel zweier ganzer Zahlen gespeichert werden, werden die Gleichungen dieses Systems mit Hilfe des Gaußschen Algorithmus gelöst. Sind die Gleichungen nicht lösbar, ist auch die rudimentär-plus Formel nicht erfüllbar. Sind die Gleichungen lösbar, so werden die Lösungen in die Ungleichungen des Systems eingesetzt, und es wird überprüft, ob Ungleichungen der Form $x \neq x$ entstehen. Ist dies der Fall, so ist die Formel wieder nicht erfüllbar. Andernfalls ist das System lösbar und die Formel somit erfüllbar.

Eine deterministische Turingmaschine $\mathbb{A}$ im Sinne von $\mathcal{R}^=$, welche SAT(RUD+ FML MIT PAR IN $\mathcal{R}^=$) in polynomieller Zeit löst, berechnet bei Eingabe einer rudimentär-plus Formel und der Parameter also das System von linearen Gleichungen und Ungleichungen und ermittelt wie oben beschrieben, ob das System lösbar und somit ob die Formel erfüllbar ist. □

Bemerkung 5.3.3 Es läßt sich sogar zeigen, daß in der Struktur $\mathcal{R}^=$ für jede natürliche Zahl i die Gleichheit

$$\Sigma_i \mathrm{P}_{\mathcal{R}^=} = \mathrm{B}\Sigma_i \mathrm{P}_{\mathcal{R}^=}$$

gilt. Hierzu siehe zum Beispiel [3].

Für die letzte Bedingung für den H_1-Standardtyp erhalten wir allerdings ein negatives Resultat, die Struktur $\mathcal{R}^=$ liefert uns auf diesem Weg keine Informationen bezüglich der P = NP-Frage. Um dies zu zeigen, benötigen wir ein einfaches Lemma.

Lemma 5.3.4 *Sei $\mathcal{V}$ ein unendlich-dimensionaler Vektorraum über dem Körper $\mathcal{K}$ und seien für $m, n \in \mathbb{N}_{>0}$ und $i \in [m]$ und $j \in [n]$ Körperelemente $\lambda_{ij} \in \mathcal{K}$ und Vektoren $a_i \in \mathcal{V}$ gegeben. Dann hat das Ungleichungssystem*

$$\begin{aligned} \lambda_{11}x_1 + \ldots + \lambda_{1n}x_n &\neq a_1 \\ &\vdots \\ \lambda_{m1}x_1 + \ldots + \lambda_{mn}x_n &\neq a_m \end{aligned}$$

eine Lösung.

Beweis: Da $\mathcal{V}$ ein unendlich-dimensionaler Vektorraum ist, gibt es linear unabhängige Vektoren $x_1, \ldots, x_n$, so daß $[\{x_1, \ldots, x_n\}]^{\mathcal{V}} \cap [\{a_1, \ldots, a_m\}]^{\mathcal{V}} = \{0\}$. Diese sind eine Lösung des Ungleichungssystems. □

Nun können wir nicht nur zeigen, daß in $\mathcal{R}^=$ die Klassen $\mathrm{P}_{\mathcal{R}^=}$ und $\mathrm{NP}_{\mathcal{R}^=}$ verschieden sind, sondern leider auch, daß es in $\mathcal{R}^=$ kein boolesches bezüglich $\mathrm{P}_{\mathcal{R}^=}$-Reduktion $\mathrm{NP}_{\mathcal{R}^=}$-vollständiges Problem gibt.

Satz 5.3.5 *In der Struktur $\mathcal{R}^=$ gilt*

(i) $\mathrm{P}_{\mathcal{R}^=} \neq \mathrm{NP}_{\mathcal{R}^=}$

(ii) es gibt kein boolesches bezüglich $\mathrm{P}_{\mathcal{R}^=}$-Reduktion $\mathrm{NP}_{\mathcal{R}^=}$-vollständiges Problem.

Beweis: *Zu (i):* Wir betrachten das Problem, bei Eingabe von $\bar{x} \in \mathbb{R}^*$ zu entscheiden, ob es ein Tupel $\bar{\varepsilon} \in \{0,1\}^{|\bar{x}|}$ mit $\varepsilon_i \neq 0$ für mindestens ein $i \in [|\bar{x}|]$ gibt, so daß $\varepsilon_1 x_1 + \ldots + \varepsilon_{|\bar{x}|} x_{|\bar{x}|} = 0$ gilt. Da wir in der Struktur $\mathcal{R}^=$ nicht über die Multiplikation verfügen, interpretieren wir die Produkte $\varepsilon_i x_i$ durch $S(\varepsilon, 0, x)$. Wir nennen dieses Problem X. Das Problem X ist in $\mathrm{BNP}_{\mathcal{R}^=}$ und somit auch in $\mathrm{NP}_{\mathcal{R}^=}$, denn eine binär nichtdeterministische Turingmaschine im Sinne von $\mathcal{R}^=$, welche bei Eingabe von $\bar{x}$ ein $\bar{\varepsilon} \in \{0,1\}^{|\bar{x}|}$ rät und überprüft, ob $\sum_{i=1}^{n} S(\varepsilon_i, 0, x_i) = 0$ gilt, akzeptiert X in polynomieller Zeit.

Um zu sehen, daß X nicht in $\mathrm{P}_{\mathcal{R}^=}$ liegt, nehmen wir an, es gäbe eine deterministische Turingmaschine $\mathbb{A}$ im Sinne von $\mathcal{R}^=$, welche X in Zeit $p(n)$ entscheidet. Die Maschine $\mathbb{A}$ rechnet bei Eingabe von $\bar{x}$ in der von $\bar{x}$ in $\mathcal{R}^=$ erzeugten Substruktur. Während der Berechnung führt $\mathbb{A}$ Tests mit der Eingabe $\bar{x}$ durch, jeder von diesen hat dabei die Form $z_1 x_1 + \ldots + z_{|\bar{x}|} x_{|\bar{x}|} = c?$, wobei $z_i \in \mathbb{Z}$ für $i \in [|\bar{x}|]$ und $c \in \mathbb{Z}$. Nach einer entsprechenden Normierung, die das Ergebnis des jeweiligen Tests nicht verändert, haben alle durchgeführten Tests die Form

$$q_1 x_1 + \ldots + q_{|\bar{x}|} x_{|\bar{x}|} = c'?$$

wobei $q_i \in \mathbb{Q}$ für $i \in [|\bar{x}|]$ und $c' \in \{0,1\}$. Da $\mathbb{A}$ das Problem X in Zeit $p(n)$ entscheidet, werden also auch höchstens $p(|\bar{x}|)$ Tests durchgeführt.

Wir wollen nun zwei Eingaben $\bar{a}$ und $\bar{b}$ mit $\bar{a} \notin X$ und $\bar{b} \in X$ konstruieren, so daß $\mathbb{A}$ bei beiden Eingaben dasselbe Ergebnis liefert. Sei $n \in \mathbb{N}$ genügend groß, so daß $p(n) < 2^n - 1$ gilt und sei $\bar{a} \in \mathbb{R}^n$, so daß $a_1, \ldots, a_n, 1$ linear unabhängig über $\mathbb{Q}$ sind. Nach den obigen Überlegungen führt $\mathbb{A}$ höchstens $p(n)$ Tests $q_1 a_1 + \ldots + q_n a_n = c'?$ durch. Aufgrund der linearen Unabhängigkeit von $(\bar{a}, 1)$ ist jeder dieser Tests negativ. Da $p(n) < 2^n - 1$, gibt es ein $\bar{\eta} \in \{0,1\}^n$ mit $\eta_i = 1$ für mindestens ein $i \in [n]$, so daß kein Test der Form $\eta_1 a_1 + \ldots + \eta_n a_n = c?$ in der Berechnung von $\mathbb{A}$ vorkommt. Sei nun $k \in [n]$ so, daß $\eta_k = 1$. Wir setzen

$$v_k := \sum_{i \in [n], i \neq k} \eta_i v_i$$

und betrachten für die Tupel $(q_{11}, \ldots, q_{1n}, c_1), \ldots, (q_{p(n)1}, \ldots, q_{p(n)n}, c_{p(n)})$, die in den Tests der Berechnung von $\mathbb{A}$ bei Eingabe von $\bar{a}$ vorkommen, das Ungleichungssystem in $n-1$ Variablen

$$\begin{aligned} q_{11} v_1 + \ldots + q_{1n} v_n &\neq c_1 \\ &\vdots \\ q_{p(n)1} v_1 + \ldots + q_{p(n)n} v_n &\neq c_{p(n)} \end{aligned}$$

Mit Lemma 5.3.4 hat das Ungleichungssystem eine Lösung $(b_1, \ldots, b_{k-1}, b_{k+1}, \ldots, b_n) \in \mathbb{R}^{n-1}$. Mit

$$b_k := \sum_{i \in [n], i \neq k} \eta_i b_i$$

erhalten wir ein $\bar{b} \in \mathbb{R}^n$, so daß $\bar{x} \in X$ gilt. Allerdings kann die Maschine $\mathbb{A}$ die béiden Eingaben $\bar{a} \notin X$ und $\bar{b} \in X$ nicht unterscheiden, da jeder durchgeführte Test dasselbe Ergebnis liefert. Somit kann $\mathbb{A}$ das Problem X nicht entscheiden.

Zu (ii): Falls es ein boolesches bezüglich $\mathrm{P}_{\mathcal{R}^=}$-Reduktion $\mathrm{NP}_{\mathcal{R}^=}$-vollständiges Problem Y gibt, so gibt es auch eine $\mathrm{P}_{\mathcal{R}^=}$-Reduktion von X auf Y. Betrachten wir eine in polynomieller Zeit berechenbare Funktion $f : \mathbb{R}^* \to \{0,1\}^*$. Da die Funktionswerte von f boolesch sind, können wir ohne Beschränkung der Allgemeinheit davon ausgehen daß die diese Funktion in Zeit $p(n)$ berechnende deterministische Turingmaschine $\mathbb{A}$ im Sinne von $\mathcal{R}^=$ zuletzt jedes Element ihrer Ausgabe daraufhin testet, ob es boolesch ist. Wie oben konstruieren wir nun für ein entsprechend großes $n \in \mathbb{N}$ zwei Eingaben $\bar{a}, \bar{b} \in \mathbb{R}^n$, mit $\bar{a} \notin X$ und $\bar{b} \in X$, so daß $\mathbb{A}$ bei Eingabe von $\bar{a}$ dasselbe Ergebnis liefert wie bei Eingabe von $\bar{b}$. Damit kann die Funktion f jedoch keine $\mathrm{P}_{\mathcal{R}^=}$-Reduktion von X auf Y sein. Somit gibt es kein boolesches bezüglich $\mathrm{P}_{\mathcal{R}^=}$-Reduktion $\mathrm{NP}_{\mathcal{R}^=}$-vollständiges Problem. □

Die Struktur der reellen Zahlen mit Addition und additiver Inversenbildung liefert uns also auf dem Wege des Standardtyps leider keine Informationen über die P = NP-Frage im Klassischen.

5.3.2 Mit Addition, additivem Inversen und Ordnung

Fügen wir zu der Struktur $\mathcal{R}^=$ noch die Ordnung hinzu, so erhalten wir die Struktur $\mathcal{R}^< := (\mathbb{R}, +, -, <)$. Die Situation ist hier ähnlich wie in der oben betrachteten Struktur $\mathcal{R}^=$.

Satz 5.3.6 *In der Struktur $\mathcal{R}^<$ gilt*

$$\mathrm{P}_{\mathcal{R}^<} \restriction \{0,1\} = \mathrm{P}.$$

Beweis: Die Inklusion $\mathrm{P} \subseteq \mathrm{P}_{\mathcal{R}^<} \restriction \{0,1\}$ gilt wieder, da die booleschen Funktionen mittels der Selektorfunktion simuliert werden können.

Um die Inklusion $\mathrm{P}_{\mathcal{R}^<} \restriction \{0,1\} \subseteq \mathrm{P}$ zu zeigen, betrachten wir wie im Beweis zu Satz 5.3.1 ein Problem $X \in \mathrm{P}_{\mathcal{R}^<} \restriction \{0,1\}$. Für ein $X' \in \mathrm{P}_{\mathcal{R}^<}$ gilt dann $X = X' \cap \{0,1\}^*$. Sei wieder $\mathbb{A}$ die deterministische Turingmaschine im Sinne von $\mathcal{R}^<$, welche das Problem X' in Zeit $p(n)$ entscheidet. In der Berechnung von $\mathbb{A}$ kommen bei Eingabe von $\bar{\varepsilon} \in \{0,1\}^*$ wieder nur ganze Zahlen z mit $|z| \leq 2^{p(n)} + 1$ vor. Die binären Kodierungen dieser Zahlen haben somit alle höchstens die Länge $\log(2^{p(n)}) + 1 = p(n) + 1$.

Für eine geeignete Konstante $C \in \mathbb{N}$ lassen sich zwei ganze Zahlen in binärer Kodierung der Länge höchstens $p(n) + 1$ in höchstens $C \cdot p(n)$ Schritten addieren oder bezüglich der Ordnung vergleichen, und das additive Inverse einer solchen Zahl läßt sich ebenso in höchstens $C \cdot p(n)$ Schritten berechnen. Eine klassische Turingmaschine $\mathbb{A}'$, welche die Maschine $\mathbb{A}$ simuliert, indem sie mit den entsprechenden Kodierungen rechnet, entscheidet also für binäre Eingaben das Problem X' und benötigt, da $\mathbb{A}$ für Eingaben der Länge n höchstens $p(n)$ Schritte läuft, höchstens $C \cdot (p(n))^2$ Schritte. Somit gilt $X \in \mathrm{P}$. □

Bemerkung 5.3.7 Wie sich im Nachhinein herausgestellt hat, ist dieses Ergebnis auch in [14] zu finden (allerdings ohne Beweis).

Hier sehen wir wieder, daß es in bezug auf die Komplexität von Berechnungen einen großen Unterschied macht, ob wir die Elemente des Universums einer Struktur als Konstanten zu dieser Struktur hinzunehmen, oder nicht:

Bemerkung 5.3.8 Es läßt sich zeigen, daß für die Struktur $(\mathcal{R}^<, (c_r)_{r\in\mathbb{R}})$ gilt:

$$\mathrm{P}_{(\mathcal{R}^<,(c_r)_{r\in\mathbb{R}})} \restriction \{0,1\} = \mathbb{P}$$

(siehe hierfür zum Beispiel [22] oder [26]). Mit Lemma 2.1.15 gilt aber $\mathrm{P} \neq \mathbb{P}$ und mit Satz 5.3.6 somit auch $\mathrm{P}_{\mathcal{R}^<} \restriction \{0,1\} \neq \mathrm{P}_{(\mathcal{R}^<,(c_r)_{r\in\mathbb{R}})} \restriction \{0,1\}$. Damit gilt aber

$$\mathrm{P}_{\mathcal{R}^<} \neq \mathrm{P}_{(\mathcal{R}^<,(c_r)_{r\in\mathbb{R}})}.$$

Andererseits ist jede deterministische Turingmaschine im Sinne von $\mathcal{R}^<$ auch eine deterministische Turingmaschine im Sinne von $(\mathcal{R}^<, (c_r)_{r\in\mathbb{R}})$, es gilt somit $\mathrm{P}_{\mathcal{R}^<} \subseteq \mathrm{P}_{(\mathcal{R}^<,(c_r)_{r\in\mathbb{R}})}$. Also gibt es ein Problem $X \subseteq \mathbb{R}^*$, welches nur unter Hinzunahme von Parametern im Sinne von $\mathcal{R}^<$ in polynomieller Zeit entschieden werden kann.

Bezüglich der zweiten Bedingung für den H_1-Standardtyp erhalten wir wie für die Struktur $\mathcal{R}^=$ ein positives Ergebnis. Der Beweis läuft ähnlich wie im vorhergehenden Fall, wird jedoch dadurch etwas komplizierter, daß wir auch noch für die Ordnung Sorge tragen müssen. Wir benötigen zunächst folgendes Lemma.

Lemma 5.3.9 (Caratheodory) *Sei $\mathcal{V}$ ein $\mathbb{R}$-Vektorraum und seien $b, a_1, \ldots, a_n$ Vektoren aus $\mathcal{V}$. Wenn es nichtnegative reelle Zahlen $\lambda_1, \ldots, \lambda_n$ gibt, so daß $b = \lambda_1 a_1 + \ldots + \lambda_n a_n$ gilt, so gibt es linear unabhängige Vektoren $a_{i_1}, \ldots, a_{i_k} \in \{a_1, \ldots, a_n\}$ und nichtnegative reelle Zahlen $\lambda'_1, \ldots, \lambda'_k$, so daß $b = \lambda'_1 a_{i_1} + \ldots + \lambda'_k a_{i_k}$ gilt.*

Beweis: Falls die Vektoren $a_1, \ldots, a_n$ linear unabhängig sind, ist nichts zu zeigen. Sind die $a_1, \ldots, a_n$ andererseits linear abhängig, so gibt es $\mu_1, \ldots, \mu_n \in \mathbb{R}$ mit $\mu_i \neq 0$ für mindestens ein $i \in [n]$ und $\mu_1 a_1 + \ldots + \mu_n a_n = 0$. Wir wählen ein $j \in [n]$, so daß $|\frac{\lambda_j}{\mu_j}|$ minimal ist. Dann gilt $a_j = (-\sum_{i \in [n], i \neq j} \mu_i a_i) \cdot \mu_j^{-1}$ und somit

$$b = \sum_{i \in [n], i \neq j} (\lambda_i - \frac{\lambda_j}{\mu_j} \cdot \mu_i) \cdot a_i.$$

Da $\lambda_i \geq 0$ für alle $i \in [n]$ gilt, und da $|\frac{\lambda_j}{\mu_j}|$ minimal ist, gilt auch $\lambda_i - \frac{\lambda_j}{\mu_j} \cdot \mu_i \geq 0$ für alle $i \in [n]$ mit $i \neq j$. Wenn wir diesen Schritt iterieren, erhalten wir schließlich die gesuchten linear unabhängigen Vektoren $a_{i_1}, \ldots, a_{i_k}$ und die nichtnegativen Koeffizienten $\lambda'_1, \ldots, \lambda'_k$. □

Mithilfe des Lemmas von Caratheodory läßt sich nun die folgende Aussage zeigen. Wir bezeichnen hierbei mit der *Länge einer ganzen Zahl* die Länge ihrer binären Kodierung.

Lemma 5.3.10 *Wenn ein System E aus n Ungleichungen*

$$\begin{aligned}
\lambda_{11} x_1 + \ldots + \lambda_{1n} x_n &\leq a_1 \\
&\vdots \\
\lambda_{k1} x_1 + \ldots + \lambda_{kn} x_n &\leq a_k \\
\mu_{11} x_1 + \ldots + \mu_{1n} x_n &< b_1 \\
&\vdots \\
\mu_{l1} x_1 + \ldots + \mu_{ln} x_n &< b_l
\end{aligned}$$

mit ganzzahligen Koeffizienten λ_{ij}, μ_{ij} der Länge n und mit $a_1, \ldots, a_k, b_1, \ldots, b_l \in \mathbb{R}$ eine Lösung hat, dann hat es auch eine Lösung der Form $(\frac{c_1}{d_1}, \ldots, \frac{c_n}{d_n})$, wobei c_i und d_i für $i \in [n]$ Linearkombinationen in $a_1, \ldots, a_k, b_1, \ldots, b_l, 1$ sind, deren Koeffizienten ganzzahlig und von in n polynomieller Länge sind.

Beweis: Sei also E ein derartiges System von Ungleichungen. Wir konstruieren zunächst ein Ungleichungssystem F, in dem keine strikten Ungleichungen mehr vorkommen, und welches genau dann lösbar ist, wenn das System E lösbar ist. Hierzu beobachten wir, daß mit jeder der Ungleichungen der Form $\mu_{i1} x_1 + \ldots + \mu_{in} x_n < b_i$ für ein hinreichend großes y die Ungleichung $\mu_{i1} x_1 \cdot y + \ldots + \mu_{in} x_n \cdot y \leq b_i \cdot y - 1$ gilt. Wir bilden also das Ungleichungssystem F:

$$\begin{aligned}
\lambda_{11} x_1 + \ldots + \lambda_{1n} x_n &\leq a_1 \cdot y \\
&\vdots
\end{aligned}$$

$$\begin{aligned}
\lambda_{k1}x_1 + \ldots + \lambda_{kn}x_n &\leq a_k \cdot y \\
\mu_{11}x_1 + \ldots + \mu_{1n}x_n &\leq b_1 \cdot y - 1 \\
&\vdots \\
\mu_{l1}x_1 + \ldots + \mu_{ln}x_n &\leq b_l \cdot y - 1 \\
1 &\leq y
\end{aligned}$$

In F kommen nun keine strikten Ungleichungen mehr vor, und das System E ist genau dann lösbar, wenn das System F lösbar ist. Um das Lemma von Caratheodory anwenden zu können, brauchen wir nun ein Gleichheitssystem, welches genau dann nichtnegativ erfüllbar ist, wenn das System F erfüllbar ist. Hierzu führen wir für jede Variable x_i von F zwei neue Variablen u_i und v_i ein. Außerdem führen wir für jede der Ungleichungen aus F eine neue Variable w_j ein. Damit erhalten wir das folgende Gleichungssystem G:

$$\begin{aligned}
\lambda_{11}(u_1 - v_1) + \ldots + \lambda_{1n}(u_n - v_n) + w_1 - a_1 y &= 0 \\
&\vdots \\
\lambda_{k1}(u_1 - v_1) + \ldots + \lambda_{kn}(u_n - v_n) + w_k - a_k y &= 0 \\
\mu_{11}(u_1 - v_1) + \ldots + \mu_{1n}(u_n - v_n) + w_{k+1} - b_1 y &= -1 \\
&\vdots \\
\mu_{l1}(u_1 - v_1) + \ldots + \mu_{ln}(u_n - v_n) + w_n - b_l y &= -1 \\
w_{n+1} - y &= -1
\end{aligned}$$

Das Gleichungssystem G hat nun genau dann eine nichtnegative Lösung, wenn das Gleichungssystem F eine Lösung hat. Die der linken Seite von G entsprechende $(n+1) \times (3n+2)$-Matrix M_G hat dann die folgende Gestalt:

$$\begin{pmatrix}
\lambda_{11} & -\lambda_{11} & \ldots & \lambda_{1n} & -\lambda_{1n} & 1 & 0 & \ldots & & 0 & -a_1 \\
\vdots & & & & \vdots & 0 & 1 & & & & \vdots \\
\lambda_{k1} & -\lambda_{k1} & \ldots & \lambda_{kn} & -\lambda_{kn} & & & & & & -a_k \\
\mu_{11} & -\mu_{11} & \ldots & \mu_{1n} & -\mu_{1n} & & & \ddots & & & -b_1 \\
\vdots & & & & \vdots & & & & & & \vdots \\
\mu_{l1} & -\mu_{l1} & \ldots & \mu_{ln} & -\mu_{ln} & & & & 1 & 0 & -b_l \\
0 & & \ldots & & 0 & 0 & & \ldots & 0 & 1 & -1
\end{pmatrix}$$

Da die Matrix M_G die $(n+1) \times (n+1)$-Einheitsmatrix enthält, hat M_G den Rang $n+1$. Wenn das Gleichungssystem G eine nichtnegative Lösung hat, gibt es mit dem Lemma von Caratheodory somit eine aus $n+1$ (als Spaltenvektoren aufgefasst linear unabhängigen) Spalten von M_G bestehende Matrix H und nichtnegative reelle Zahlen $x_1, \ldots, x_{n+1}$, so daß gilt

$$H \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_{n+1} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ \vdots \\ -1 \end{pmatrix}.$$

Da die $x_1, \ldots, x_{n+1}$ nichtnegativ sind, muß die letzte Spalte von M_G auch eine Spalte von H sein, da andernfalls die letzte Zeile nicht erfüllbar ist. Mit der Cramerschen Regel gilt nun für jede Komponente der Lösung $x_1, \ldots, x_{n+1}$ die Gleichung

$$x_j = \frac{\det H_j}{\det H},$$

wobei H_j diejenige Matrix ist, welche aus H entsteht, wenn die j-te Spalte durch den Vektor ${}^t(\overbrace{0, \ldots, 0}^{k}, \overbrace{-1, \ldots, -1}^{l})$ ersetzt wird. Da mit der Leibniz-Formel jede dieser Determinanten die Form $\sum_{\sigma \in S(n+1)} \operatorname{sgn}(\sigma) \cdot z_{1\sigma(1)} \cdot \ldots \cdot z_{(n+1)\sigma(n+1)}$ hat, also eine Summe aus $(n+1)!$ Produkten von je $(n+1)$ Zahlen ist, da die Koeffizienten λ_{ij} und μ_{ij} ganzzahlig sind, und da in einem dieser Produkte jeweils nur eines der a_i oder eines der b_i vorkommt, sind die Determinanten jeweils Linearkombinationen aus $1, a_1, \ldots, a_k, b_1, \ldots, b_l$. Somit hat die Lösung $x_1, \ldots, x_{n+1}$ schon die richtige Gestalt.

Da die Länge der λ_{ij} und der μ_{ij} nach Voraussetzung durch n beschränkt ist, sind die Koeffizienten in den Linearkombinationen betragsmäßig kleiner als $(n+1)! \cdot (n+1) \cdot n$ und somit, da $(n+1)! \leq (n+1)^{(n+1)}$ für hinreichend großes n gilt, auch kleiner als $n(n+1)^{(n+2)}$. Die Länge der Koeffizienten beträgt also höchstens $\log n + (n+2) \cdot \log (n+1) + 1$.

Aus dieser Lösung erhalten wir nun eine nichtnegative Lösung des Gleichungssystems G, indem wir die Variablen, die vorher durch das Streichen der Spalten weggefallen sind, gleich null setzen. Daraus erhalten wir dann eine entsprechende Lösung für das System F und schließlich auch eine Lösung für das ursprüngliche System E. Diese hat dann die Form $(\frac{c_1}{d_1}, \ldots, \frac{c_n}{d_n})$, wobei die $c_1, \ldots, c_n, d_1, \ldots, d_n$ Linearkombinationen in $1, a_1, \ldots, a_k, b_1, \ldots, b_l$ sind. Da hierbei nur einmal zwei Variablen miteinander multipliziert werden, bleibt die Länge der Koeffizienten der Lösung polynomiell beschränkt. □

Mit diesem etwas aufwendigeren Lemma läßt sich der ersehnte Satz beweisen.

Satz 5.3.11 *In der Struktur $\mathcal{R}^<$ gilt*

$$\mathrm{NP}_{\mathcal{R}^<} = \mathrm{BNP}_{\mathcal{R}^<}.$$

Beweis: Um die Behauptung zu zeigen, zeigen wir

$$\textsc{Sat}(\textsc{Rud}+ \textsc{Fml mit Par in } \mathcal{R}^<) \in \mathrm{BNP}_{\mathcal{R}^<}.$$

Mit Lemma 4.2.18 gilt dann $\mathrm{NP}_{\mathcal{R}^<} = \mathrm{BNP}_{\mathcal{R}^<}$.

Eine rudimentär-plus Formel mit Parametern läßt sich mit einem $\mathrm{BNP}_{\mathcal{R}^<}$-Algorithmus in ein Ungleichungssystem der Form

$$\begin{aligned}
\lambda_{11}x_1 + \ldots + \lambda_{1n}x_n &\leq a_1 \\
&\vdots \\
\lambda_{k1}x_1 + \ldots + \lambda_{kn}x_n &\leq a_k \\
\mu_{11}x_1 + \ldots + \mu_{1n}x_n &< b_1 \\
&\vdots \\
\mu_{l1}x_1 + \ldots + \mu_{ln}x_n &< b_l
\end{aligned}$$

mit ganzzahligen Koeffizienten λ_{ij} und μ_{ij} von polynomiell beschränkter Länge und reellen Zahlen $a_1, \ldots, a_k, b_1, \ldots, b_l$ umwandeln. Hierzu werden zunächst in den einzelnen Konjunkten die Parameter eingesetzt. Dann werden die einzelnen Konjunkte in Zeilen des Ungleichungssystems übersetzt. So wird zum Beispiel ein Konjunkt $x_i = x_j$ durch die beiden Zeilen

$$\begin{aligned} x_i - x_j &\leq 0 \\ x_j - x_i &\leq 0 \end{aligned}$$

wiedergegeben. Das Konjunkt $x_i + a = x_j$ mit dem eingesetzten Parameter a wird durch die beiden Zeilen

$$\begin{aligned} x_i - x_j &\leq -a \\ x_j - x_i &\leq a \end{aligned}$$

wiedergegeben. Die übrigen Konjunkte werden analog umgewandelt. Vorsicht ist allerdings bei einem Konjunkt der Form $\neg x_i = x_j$ geboten. Hier muß zunächst geraten werden, welche der beiden mögliche Zeilen $x_i - x_j < 0$ oder $x_j - x_i < 0$ in das System geschrieben werden soll.

Mit dem vorhergehenden Lemma 5.3.10 hat das so konstruierte Ungleichungssystem genau dann eine Lösung, wenn es eine Lösung der Form $(\frac{c_1}{d_1}, \ldots, \frac{c_n}{d_n})$ hat, wobei $c_1, \ldots, c_n, d_1, \ldots, d_n$ Linearkombinationen von $a_1, \ldots, a_k, b_1, \ldots, b_l, 1$ mit ganzzahligen Koeffizienten polynomiell beschränkter Länge sind. Ein $\mathrm{BNP}_{\mathcal{R}^<}$-Algorithmus, welcher bei Eingabe einer rudimentär-plus Formel mit Parametern ein entsprechendes Ungleichungssystem konstruiert, die Koeffizienten rät und überprüft, ob die geratenen Koeffizienten eine Lösung des Gleichungssystems liefern, arbeitet somit in polynomieller Zeit und akzeptiert gerade das Problem SAT(RUD+ FML MIT PAR IN $\mathcal{R}^<$). □

Bemerkung 5.3.12 Auch hier läßt sich wieder zeigen, daß für jede natürliche Zahl i die Gleichheit

$$\Sigma_i \mathrm{P}_{\mathcal{R}^<} = \mathrm{B}\Sigma_i \mathrm{P}_{\mathcal{R}^<}$$

gilt. Siehe wieder [3].

Betrachten wir statt der Struktur $\mathcal{R}^<$ die Struktur $\mathcal{Q}^< = (\mathbb{Q}, +, -, <)$, so erhalten wir aus dem Beweis des vorherigen Satzes das folgende Korollar.

Korollar 5.3.13 *Für* $\mathcal{Q}^< = (\mathbb{Q}, +, -, <)$ *gilt*

(i) $\mathrm{P}_{\mathcal{Q}^<} \restriction \{0,1\} = \mathrm{P}$

(ii) $\mathrm{NP}_{\mathcal{Q}^<} = \mathrm{BNP}_{\mathcal{Q}^<}$

Beweis: *Zu (i):* Zunächst gilt offensichtlicherweise $\mathrm{P} \subseteq \mathrm{P}_{\mathcal{Q}^<} \restriction \{0,1\}$. Da $\mathcal{Q}^<$ Substruktur von $\mathcal{R}^<$ ist, gilt andererseits auch $\mathrm{P}_{\mathcal{Q}^<} \restriction \{0,1\} \subseteq \mathrm{P}_{\mathcal{R}^<} \restriction \{0,1\}$. Mit Lemma 5.3.6 ist jedoch $\mathrm{P}_{\mathcal{R}^<} \restriction \{0,1\} = \mathrm{P}$ und somit auch $\mathrm{P}_{\mathcal{Q}^<} \restriction \{0,1\} = \mathrm{P}$.

Zu (ii): Analog zum Beweis von Satz 5.3.11. □

Für die Struktur $\mathcal{R}^<$ ist leider nicht bekannt, ob es ein boolesches bezüglich $\mathcal{R}^<$-Reduktion $\mathrm{NP}_{\mathcal{R}^<}$-vollständiges Problem gibt. Es ist also nicht bekannt, ob $\mathcal{R}^<$ vom H_1-Standardtyp ist. Dennoch läßt sich mittels Turingmaschinen, welche Zugriff auf ein Orakel haben, zeigen, daß

$$\mathrm{P}_{\mathcal{R}^<} = \mathrm{NP}_{\mathcal{R}^<} \Leftrightarrow \mathrm{P} = \mathrm{NP}$$

gilt (siehe hierzu [14]).

Nimmt man zu der Struktur $\mathcal{R}^<$ noch die Multiplikation hinzu, so erhält man den angeordneten Körper der Reellen Zahlen. Körper im Allgemeinen werden wir in Abschnitt 5.4 betrachten.

5.3.3 $\mathbb{R}$ mit Addition, additivem Inversen, Multiplikation und Ordnung

Wir betrachten nun die Struktur $\mathcal{R} = (\mathbb{R}, +, -, \cdot, <)$, also die reellen Zahlen mit der Addition, der additiven Inversenbildung, der Multiplikation und der Ordnung. Zwar erlaubt die Struktur $\mathcal{R}$ Quantorenelimination, es ist jedoch leider nicht bekannt, ob $\mathrm{NP}_{\mathcal{R}} = \mathrm{BNP}_{\mathcal{R}}$ gilt. Wir können dagegen zwei negative Ergebnisse zeigen.

Satz 5.3.14 *In der Struktur $\mathcal{R}$ gilt:*

(i) es gibt ein boolesches nicht im Sinne von $\mathcal{R}$ entscheidbares Problem

(ii) es gibt ein boolesches Problem, welches nicht in $\mathbb{P}_{\mathcal{R}}$ liegt.

Beweis: *Zu (i):* Da die Symbolmenge von $\mathcal{R}$ endlich ist, können deterministische Turingmaschinen im Sinne von $\mathcal{R}$ binär kodiert werden. Somit gibt es höchstens $\aleph_0$ viele derartige Maschinen. Dagegen gibt es überabzählbar viele boolesche Probleme. Somit gibt es ein boolesches Problem, welches nicht im Sinne von $\mathcal{R}$ entscheidbar ist.

Zu (ii): Wir zeigen wieder, daß es mehr boolesche Probleme gibt als von Schaltkreisfamilien polynomieller Größe im Sinne von $\mathcal{R}$ akzeptiert werden können. Hierzu konstruieren wir für jede positive natürliche Zahl k eine natürliche Zahl n_k, so daß es ein Problem $X_k \in \{0,1\}^{n_k}$ gibt, welches von keinem Schaltkreis der Größe $(n_k)^k$ akzeptiert wird. Das Problem $Y := \bigcup_{k \in \mathbb{N}_{>0}} X_k$ wird dann von keiner Schaltkreisfamilie polynomieller Größe akzeptiert, denn wenn das Polynom, welches die Größe der Schaltkreise beschränkt, den Grad k hat, so wird nach Konstruktion von Y und n_k das Problem $Y \cap \{0,1\}^{n_k}$ nicht von dem Schaltkreis C_{n_k} akzeptiert.

Betrachten wir also die Anzahl der von Schaltkreisen der Größe höchstens n^k bei boolescher Eingabe berechneten Funktionen. Diese ist dieselbe wie die Anzalh der von Schaltkreisen der Größe höchstens n^k, deren Knoten nicht mit Variablen gelabelt sind, berechneten Funktionen. Wir wollen die Anzahl der Schaltkreise der Größe höchstens n^k, deren Knoten nicht mit Variablen gelabelt sind, berechnen.

Ein Schaltkreis der Größe n^k hat n^k Knoten, welche jeweils mit einem der Symbole $+, -, \cdot, <, =, 0, 1, \mathrm{Id}, S$ gelabelt sind. Da wir auch Schaltkreise der Größe $n' < n^k$ betrachten müssen, fügen wir ein weiteres Symbol hinzu, welches kennzeichnet, daß der betreffende Knoten nicht Teil des Schaltkreises ist. Somit gibt es für das Label jedes Knotens

10 Möglichkeiten. Insgesamt gibt es also $10^{(n^k)} = 2^{(n^k)\log 10}$ Möglichkeiten, die Knoten zu labeln. Da an jedem der Knoten höchstens drei Pfeile enden, gibt es zwischen je zwei Knoten entweder keinen, einen, zwei oder drei Pfeile. Da es $(n^k)^2 = n^{2k}$ Paare von Knoten gibt, gibt es somit $4^{(n^{2k})} = 2^{(n^{2k})\log 4}$ Möglichkeiten für die Pfeile. Insgesamt gibt es also höchstens $2^{(n^{2k})\log 4+(n^k)\log 10}$ Schaltkreise der Größe n^k.

Allerdings gibt es 2^{2^n} Funktionen von $\{0,1\}^n$ nach $\{0,1\}$. Da $2^{(n^{2k})\log 4+(n^k)\log 10} < 2^{2^n}$ für hinreichend große n gilt, gibt es für jedes k eine natürliche Zahl n_k, so daß es mehr Funktionen von $\{0,1\}^{n_k}$ nach $\{0,1\}$ gibt, als Schaltkreise der Größe n_k^k. Es gibt somit für jedes $k \in \mathbb{N}$ ein Problem $X_k \subseteq \{0,1\}^{n_k}$, welches von keinem Schaltkreis der Größe $(n_k)^k$ akzeptiert wird. □

Bemerkung 5.3.15 Auch hier erhalten wir für die mit Konstanten für jedes Element des Universums verlängerte Struktur $(\mathcal{R}, (c_r)_{r\in\mathbb{R}})$ teilweise ein anderes Ergebnis. So läßt sich zeigen, daß jedes boolesche Problem im Sinne von $(\mathcal{R}, (c_r)_{r\in\mathbb{R}})$ in exponentieller Zeit entscheidbar ist (siehe [22]). Dies ist analog zum Beweis von Satz 2.1.16 dadurch möglich, daß unendliche 0-1-Folgen in einzelne reelle Zahlen kodiert werden können.

5.4 Körper

Als letzte Beispiele wollen wir Körper betrachten. In Körpern ist es möglich, die Selektorfunktion zu simulieren, indem bei Eingabe von x_1, x_2, x_3 getestet wird, ob die Eingabe x_1 binär ist, falls ja das richtige Ergebnis mittels des Terms $(1 - x_1) \cdot x_2 + x_1 \cdot x_3$ ausgegeben wird, andernfalls x_1. Wir verzichten deswegen in diesem Abschnitt ausnahmsweise auf die Bedingung, daß die betrachteten Strukturen eine Selektorfunktion besitzen.

Es ist leider nicht bekannt, ob in Körpern generell $\mathrm{NP}_\mathcal{K} = \mathrm{BNP}_\mathcal{K}$ gilt, oder ob es ein boolesches $\mathrm{NP}_\mathcal{K}$-vollständiges Problem gibt. Allerdings läßt sich in Körpern der Charakteristik ungleich 0 die Gleichheit $\mathrm{P}_\mathcal{K} \restriction \{0,1\} = \mathrm{P}$ zeigen:

Satz 5.4.1 *In einem Körper $\mathcal{K}$ mit $char(\mathcal{K}) \neq 0$ gilt*

- $\mathrm{P}_\mathcal{K} \restriction \{0,1\} = \mathrm{P}$
- $\mathbb{P}_\mathcal{K} \restriction \{0,1\} = \mathbb{P}$

Beweis Sei $\mathcal{K}$ also ein Körper der Charakteristik p mit $p \neq 0$. Es ist klar, daß $\mathrm{P} \subseteq \mathrm{P}_\mathcal{K} \restriction \{0,1\}$ gilt. Da andererseits $\mathcal{K}$ die Charakteristik p hat, ist $\mathbb{F}_p$ (modulo Isomorphie) ein Unterkörper von $\mathcal{K}$. Es gilt also mit Lemma 4.3.2 die Gleichheit

$$\mathrm{P}_\mathcal{K} \restriction \{0,1\} = \mathrm{P}_{\mathbb{F}_p} \restriction \{0,1\}.$$

Da der Körper $\mathbb{F}_p$ endlich ist, gilt mit Lemma 5.0.10 allerdings $\mathrm{P}_{\mathbb{F}_p} \restriction \{0,1\} = \mathrm{P}$ und somit auch $\mathrm{P}_\mathcal{K} \restriction \{0,1\} = \mathrm{P}$. Im nicht uniformen Fall analog. □

Für Körper der Charakteristik 0 ist für den uniformen Fall ein analoges Ergebnis leider nicht bekannt. Im nicht uniformen Fall läßt sich jedoch auch hier die Gleichheit $\mathbb{P}_\mathcal{K} \restriction \{0,1\} = \mathbb{P}$ beweisen. Wir betrachten dazu zunächst den Körper $\mathcal{Q}$ der rationalen

Zahlen. Ein Schaltkreis im Sinne von $\mathcal{Q}$ berechnet bei Eingabe von $\bar{\varepsilon} \in \{0,1\}^*$ in jedem seiner Knoten eine ganze Zahl. Leider können die berechneten Zahlen sehr groß werden, weswegen es nicht möglich ist, einen solchen Schaltkreis „straight forward“ mit einem klassischen Schaltkreis polynomieller Größe zu simulieren. Allerdings können wir uns mit dem Trick behelfen, den Schaltkreis modulo einer geeigneten Primzahl zu simulieren. Wir benötigen hierfür folgendes Lemma.

Lemma 5.4.2 *Sei N_t die Menge der Zahlen, die von Schaltkreisen im Sinne von $\mathcal{Q}$ der Größe höchstens t, deren Eingangsknoten nicht mit Variablen gelabelt sind, berechnet werden. Dann gibt es eine Primzahl p_t mit $\log(p_t) \leq 2t^2$, welche kein $n \in N_t$ teilt.*

Beweis: Sei N_t also wie gefordert. Wir überlegen zunächst analog zum Beweis von Satz 5.3.14, *ii)*, wie viele Schaltkreise im Sinne von $\mathcal{Q}$ der Größe t, deren Eingangsknoten nicht mit Variablen gelabelt sind, es gibt. Dann ist $|N_t|$ höchstens so groß wie diese Anzahl.

Ein solcher Schaltkreis der Größe t hat t Knoten. Da die Funktionen und Relationen von $\mathcal{Q}$ nur höchstens 2 Argumente haben, hat jeder Knoten höchstens 2 eingehende Kanten. Es gibt t^2 mögliche Paare von Knoten in einem solchen Schaltkreis, diese sind entweder mit keinem, mit einem oder mit zwei Pfeilen verbunden. Somit gibt es $3^{t^2} = 2^{(t^2 \log 3)}$ Möglichkeiten für die Pfeile. Die Knoten sind jeweils mit einem der 6 Symbole $0, 1, +, -, \cdot, =$ gelabelt, es gibt also $6^t = 2^{(t \log 6)}$ Möglichkeiten für die Label der Knoten. Insgesamt gibt es somit höchstens $2^{(t^2 \log 3 + t \log 6)}$ mögliche Schaltkreise der Größe t. Es gilt also $|N_t| \leq 2^{(t^2 \log 3 + t \log 6)}$.

Ein solcher Schaltkreis der Größe t berechnet eine ganze Zahl z mit $|z| \leq 2^t$. Somit hat jedes der n aus N_t auch höchstens 2^t Primteiler. Alle Zahlen aus N_t zusammengenommen haben also höchstens

$$2^t \cdot 2^{(t^2 \log 3 + t \log 6)} = 2^{(t^2 \log 3 + t \log 12)}$$

Primteiler.

Mit $\pi(a) := |\{p \mid p \text{ Primzahl }, p \leq a\}|$ gilt nach dem Primzahlsatz

$$\frac{\pi(a)}{\frac{a}{\ln a}} \to 1 \text{ für } a \to \infty$$

(siehe zum Beispiel [8]). Somit gibt es eine Konstante A, so daß gilt

$$\pi(2^{(2t^2)}) \geq \frac{A \cdot 2^{(2t^2)}}{t^2}.$$

Andererseits gilt für hinreichend große t

$$\frac{A \cdot 2^{(2t^2)}}{t^2} > 2^{(t^2 \log 3 + t \log 12)}.$$

Somit gibt es mehr Primzahlen p mit $p \leq 2^{(2t^2)}$ und somit auch mit $\log p \leq 2t^2$ als Primteiler von Zahlen aus N_t. Es gibt somit eine Primzahl p_t mit $\log p_t \leq 2t^2$, welche keine der Zahlen aus N_t teilt. □

Damit folgt das Ergebnis für den Körper $\mathcal{Q}$:

Satz 5.4.3 *Für den Körper $\mathcal{Q} = (\mathbb{Q}, +, -, \cdot)$ gilt*

$$\mathbb{P}_{\mathcal{Q}} \restriction \{0,1\} = \mathbb{P}.$$

Beweis: Da die booleschen Funktionen mit den Funktionen aus $\mathcal{Q}$ simuliert werden können, gibt es für jeden Schaltkreis im Sinne der Standardstruktur einen Schaltkreis im Sinne von $\mathcal{Q}$, welcher um einen linearen Faktor größer ist und bei Eingaben aus $\{0,1\}^*$ den ursprünglichen Schaltkreis simuliert. Somit gilt $\mathbb{P} \subseteq \mathbb{P}_{\mathcal{Q}}\{0,1\}$.

Andererseits ist ein Problem in der Klasse $\mathbb{P}_{\mathcal{Q}}$, wenn es eine gesetzte Familie $(C_n)_{n\in\mathbb{N}_{>0}}$ von Schaltkreisen polynomieller Größe im Sinne von $\mathcal{Q}$ gibt, welche das Problem akzeptiert. Ein solcher Schaltkreis C_n der Größe t berechnet bei Eingabe von $\varepsilon \in \{0,1\}^n$ in jedem seiner Knoten eine ganze Zahl z mit $|z| \leq 2^t$. Mit dem vorhergehenden Lemma 5.4.2 gibt es eine Primzahl p_t, deren binäre Kodierung höchstens die Länge $2t^2$ hat und die keine der möglicherweise berechneten Zahlen teilt.

Wir konstruieren eine Schaltkreisfamilie $(C'_n)_{n\in\mathbb{N}_{>0}}$ im Sinne der Standardstruktur, indem wir den Schaltkreis C'_n den Schaltkreis C_n simulieren lassen, wobei nach jedem simulierten Knoten von C_n das Ergebnis modulo der Primzahl p_n berechnet und mit diesem weitergerechnet wird. Da somit alle vorkommenden Zahlen kleiner als p_n^2 sind, hat der Schaltkreis C'_n polynomielle Größe in n. Da die so konstruierten Schaltkreise eine boolesche Eingabe genau dann akzeptieren, wenn die ursprünglichen Schaltkreise sie akzeptieren, akzeptiert die Familie $(C_n)_{n\in\mathbb{N}_{>0}}$ das entsprechende Problem eingeschränkt auf $\{0,1\}$. Dieses ist somit in der Klasse $\mathbb{P}$. Es gilt also $\mathbb{P}_{\mathcal{Q}} \restriction \{0,1\} = \mathbb{P}$. □

Da ein Körper der Charakteristik 0 einen zum Körper $\mathcal{Q}$ isomorphen Unterkörper enthält, folgt mit diesem Satz das gewünschte Ergebnis:

Korollar 5.4.4 *In jedem Körper $\mathcal{K}$ der Charakteristik 0 gilt*

$$\mathbb{P}_{\mathcal{K}} \restriction \{0,1\} = \mathbb{P}.$$

Beweis: Wenn ein Körper $\mathcal{K}$ die Charakteristik 0 hat, so gibt es einen zum Körper $\mathcal{Q}$ isomorphen Unterkörper $\mathcal{Q}'$ von $\mathcal{K}$. Mit Lemma 4.3.2 gilt also $\mathbb{P}_{\mathcal{K}} \restriction \mathbb{Q}' = \mathbb{P}_{\mathcal{Q}'}$ und somit auch

$$\mathbb{P}_{\mathcal{K}} \restriction \{0,1\} = \mathbb{P}_{\mathcal{Q}'} \restriction \{0,1\}.$$

Mit Satz 5.4.3 folgt also $\mathbb{P}_{\mathcal{K}} \restriction \{0,1\} = \mathbb{P}$. □

Es ist leider kein Algorithmus bekannt, welcher die Primzahlen p_n aus Lemma 5.4.2 in polynomieller Zeit berechnet. Es ist also nicht bekannt, ob ein analoger Satz im uniformen Fall gilt.

Anhang A

Zum Beweis von Satz 3.1.7

Beispielhafte Programmteile für die verschiedenen Knoten:

- Falls η der Eingabeknoten ist:

  Kopiere die Eingabe auf Band 2 und schreibe auf Band drei in soviele Felder, wie die Eingabe lang ist, eine 1:
  (1,`(compute Id, 1 to 2)`)
  (2,`(move 1 right)`)
  (3,`(move 2 right)`)
  (4,`(compute 1,1 to 3)`)
  (5,`(move 3 right)`)
  (6,`(if` $1 = \lambda$ `then 7, else 1)`)

  Setze Band 2 zurück:
  (7,`(move 3 left)`)
  (8,`(if` $2 = \lambda$ `then 9, else 7)`)

  Kopiere die Einsen von Band 3 auf Band 2:
  (9,`(move 3 left)`)
  (10,`(if` $3 = \lambda$ `then 14, else 11)`)
  (11,`(compute 1,1 to 2)`)
  (12,`(move 2 left)`)
  (13,`(if` $1 = \lambda$ `then 9, else 9)`)

  Setze den Zeiger von Band 2 an die richtige Position:
  (14,`(move 3 right)`)
  (15,`(if` $3 = \lambda$ `then` $q_{\beta(\eta)}$`, else 16)`)
  (16,`(move 2 right)`)
  (17,`(if` $1 = \lambda$ `then 14, else 14)`)

- Falls η mit σ_l gelabelt ist:

  Verschiebe den Zeiger von Band 2 um eine Position nach rechts:
  (q_η,`(move 2 right)`)

$(q_\eta + 1,$`(if` $1 = \lambda$ `then` $q_{\beta(\eta)}$`, else` $q_{\beta(\eta)}$`)`$)$

- Falls η mit σ_r gelabelt ist:

  Verschiebe den Zeiger von Band 2 um eine Position nach links:
  $(q_\eta,$`(move 2 left)`$)$
  $(q_\eta + 1,$`(if` $1 = \lambda$ `then` $q_{\beta(\eta)}$`, else` $q_{\beta(\eta)}$`)`$)$

- Falls η mit $x_i \leftarrow f(x_{j_1}, \ldots, x_{j_n})$ gelabelt ist:

  Kopiere die Argumente von f auf Band 3:
  $(q_\eta,$`(move 2 right)`$)$
  $(q_\eta + 1,$`(move 2 right)`$)$
  $\ldots$
  $(q_\eta + (j_1 - 1),$`(move 2 right)`$)$
  $(q_\eta + j_1,$`(compute Id, 2 to 3)`$)$
  $(q_\eta + (j_1 + 1),$`(move 3 right)`$)$
  $(q_\eta + (j_1 + 2),$`(move 2 left)`$)$
  $(q_\eta + (j_1 + 3),$`(move 2 left)`$)$
  $\ldots$
  $(q_\eta + (2j_1 + 1),$`(move 2 left)`$)$
  $(q_\eta + (2j_1 + 2),$`(move 2 right)`$)$
  $(q_\eta + (2j_1 + 3),$`(move 2 right)`$)$
  $\ldots$
  $(q_\eta + (2j_1 + j_2 + 1),$`(move 2 right)`$)$
  $(q_\eta + (2j_1 + j_2 + 2),$`(compute Id, 2 to 3)`$)$
  $(q_\eta + (2j_1 + j_2 + 3),$`(move 3 right)`$)$
  $(q_\eta + (2j_1 + j_2 + 4),$`(move 2 left)`$)$
  $(q_\eta + (2j_1 + j_2 + 5),$`(move 2 left)`$)$
  $\ldots$
  $(q_\eta + (2j_1 + 2j_2 + 3),$`(move 2 left)`$)$

  $\vdots$

  $(q_\eta + (\sum_{k=1}^{n-1} 2j_k + 2(n-1)),$`(move 2 right)`$)$
  $(q_\eta + (\sum_{k=1}^{n-1} 2j_k + 2(n-1) + 1),$`(move 2 right)`$)$
  $\ldots$
  $(q_\eta + (\sum_{k=1}^{n-1} 2j_k + 2(n-1) + (j_n - 1)),$`(move 2 right)`$)$
  $(q_\eta + (\sum_{k=1}^{n-1} 2j_k + 2(n-1) + j_n),$`(compute Id, 2 to 3)`$)$
  $(q_\eta + (\sum_{k=1}^{n-1} 2j_k + 2(n-1) + j_n + 1),$`(move 3 right)`$)$
  $(q_\eta + (\sum_{k=1}^{n-1} 2j_k + 2(n-1) + j_n + 2),$`(move 2 left)`$)$
  $(q_\eta + (\sum_{k=1}^{n-1} 2j_k + 2(n-1) + j_n + 3),$`(move 2 left)`$)$
  $\ldots$
  $(q_\eta + (\sum_{k=1}^{n-1} 2j_k + 2(n-1) + 2j_n + 1),$`(move 2 left)`$)$

  Berechne den Funktionswert:

$(q_\eta + (\sum_{k=1}^{n} 2j_k + 2n)$,(move 3 left))
$(q_\eta + (\sum_{k=1}^{n} 2j_k + 2n + 1)$,(move 3 left))
...
$(q_\eta + (\sum_{k=1}^{n} 2j_k + 2n + n)$,(move 3 left))
$(q_\eta + (\sum_{k=1}^{n} 2j_k + 2n + n + 1)$,(compute f,3 to 3))

Schreibe das Ergebnis an die entsprechende Stelle von Band 2:
$(q_\eta + (\sum_{k=1}^{n} 2j_k + 3n + 2)$,(move 2 right))
$(q_\eta + (\sum_{k=1}^{n} 2j_k + 3n + 3)$,(move 2 right))
...
$(q_\eta + (\sum_{k=1}^{n} 2j_k + 3n + (i+1))$,(move 2 right))
$(q_\eta + (\sum_{k=1}^{n} 2j_k + 3n + i + 2)$,(compute Id,3 to 2))
$(q_\eta + (\sum_{k=1}^{n} 2j_k + 3n + i + 3)$,(move 2 left))
$(q_\eta + (\sum_{k=1}^{n} 2j_k + 3n + i + 4)$,(move 2 left))
...
$(q_\eta + (\sum_{k=1}^{n} 2j_k + 3n + 2i + 2)$,(move 2 left))

Springe in die zum nächsten Knoten gehörige Zeile:
$(q_\eta + (\sum_{k=1}^{n} 2j_k + 3n + 2i + 3)$,(if $1 = \lambda$ then $q_{\beta(\eta)}$, else $q_{\beta(\eta)}$))

- Falls η mit $Rx_{i_1} \dots x_{i_n}?$ gelabelt ist:

  Kopiere die entsprechenden Feldinhalte auf Band 3:
  $(q_\eta$,(move 2 right))
  ...
  $(q_\eta + (i_1 - 1)$,(move 2 right))
  $(q_\eta + i_1$,(compute Id,2 to 3))
  $(q_\eta + (i_1 + 1)$,(move 3 right))
  $(q_\eta + (i_1 + 2)$,(move 2 left))
  ...
  $(q_\eta + (2i_1 + 1)$,(move 2 left))

  ⋮

  $(q_\eta + (\sum_{k=1}^{n-1} 2i_k + 2(n-1) - 1)$,(move 2 right))
  ...
  $(q_\eta + (\sum_{k=1}^{n-1} 2i_k + 2(n-1) + i_n - 1)$,(move 2 right))
  $(q_\eta + (\sum_{k=1}^{n-1} 2i_k + 2(n-1) + i_n)$,(compute Id,2 to 3))
  $(q_\eta + (\sum_{k=1}^{n-1} 2i_k + 2(n-1) + i_n + 1)$,(move 3 right))
  $(q_\eta + (\sum_{k=1}^{n-1} 2i_k + 2(n-1) + i_n + 2)$,(move 2 left))
  ...
  $(q_\eta + (\sum_{k=1}^{n} 2i_k + 2n - 1)$,(move 2 left))

  Teste, ob die Relation auf die entsprechenden Elemente zutrifft und springe in die entsprechende Zeile:
  $(q_\eta + (\sum_{k=1}^{n} 2i_k + 2n)$,(if $R, 3$ then $q_{\beta^+(\eta)}$, else $q_{\beta^-(\eta)}$))

- Falls η der Ausgabeknoten ist:

  Teste, ob die Ausgabe leer ist:
  $(q_\eta,$`(if` $2 = \lambda$ `then` $q_\eta + 26$`, else` $q_\eta + 1$`)`$)$

  Berechne, wie lang die Ausgabe ist:
  $(q_\eta + 1,$`(compute 0 to 3)`$)$
  $(q_\eta + 2,$`(move 2 left)`$)$
  $(q_\eta + 3,$`(if` $x, 2 = 1$ `then` $q_\eta + 4$`, else` $q_\eta + 8$`)`$)$
  $(q_\eta + 4,$`(compute 1 to 3)`$)$
  $(q_\eta + 5,$`(move 2 left)`$)$
  $(q_\eta + 6,$`(move 3 left)`$)$
  $(q_\eta + 7,$`(if` $1 = \lambda$ `then` $q_\eta + 3$`, else` $q_\eta + 3$`)`$)$

  Kopiere die Ausgabe auf Band 4:
  $(q_\eta + 8,$`(write 0 to 3)`$)$
  $(q_\eta + 9,$`(move 2 right)`$)$
  $(q_\eta + 10,$`(move 3 right)`$)$
  $(q_\eta + 11,$`(if` $x, 2 = 1$ `then` $q_\eta + 9$ `else` $q_\eta + 12$`)`$)$
  $(q_\eta + 12,$`(move 2 right)`$)$
  $(q_\eta + 13,$`(move 2 right)`$)$
  $(q_\eta + 14,$`(compute Id,` 2 `to 4)`$)$
  $(q_\eta + 15,$`(move 4 right)`$)$
  $(q_\eta + 16,$`(move 3 left)`$)$
  $(q_\eta + 17,$`(if` $x, 3 = 1$ `then` $q_\eta + 13$`, else` $q_\eta + 18$`)`$)$
  $(q_\eta + 18,$`(move 2 right)`$)$
  $(q_\eta + 19,$`(compute Id,2 to 4)`$)$

  Setze den Zeiger von Band 4 an den Beginn der Ausgabe und halt:
  $(q_\eta + 20,$`(move 4 left)`$)$
  $(q_\eta + 21,$`(move 3 left)`$)$
  $(q_\eta + 22,$`(if` $x, 4 = 1$ `then` $q_\eta + 23$`, else` $q_\eta + 25$`)`$)$
  $(q_\eta + 23,$`(move 4 left)`$)$
  $(q_\eta + 24,$`(if` $1 = \lambda$ `then` $q_\eta + 22$`, else` $q_\eta + 22$`)`$)$
  $(q_\eta + 25,$`(move 4 left)`$)$
  $(q_\eta + 26,$`(stop)`$)$

Literaturverzeichnis

[1] Arora, S.; Barak, B: *Computational Complexity: A Modern Approach* (Web Draft). http://www.cs.princeton.edu/theory/complexity/ (11.9.2007)

[2] Blum, L.; Shub, M.; Smale, S.: *On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines*. In: Bulletin of the American Mathematical Society, Vol. 21, Number 1, July 1989

[3] Blum, L.; Cucker, F.; Shub, M.; Smale, S.: *Complexity and Real Computation*. Springer, New York. 1998.

[4] Bournez, O.; Cucker, F.; de Naurois, P.; Marion, J.-Y.: *Computability over an Arbitrary Structure. Sequential and Parallel Polynomial Time*. In: A.D. Gordon (Ed.): FOSSACS 2003, LNCS 2620, 2003. S. 185-199.

[5] Böhm, C.; Jacopini, G.: *Flow Diagrams, Turing Machines And Languages With Only Two Formation Rules*. In: Computational Linguistics Vol. 9, No. 5, 1966. S.366-371.

[6] Bröcker, T.: *Lineare Algebra und Analytische Geometrie*. Birkhäuser Verlag, Basel. 2003.

[7] Bruno, J.; Steiglitz, K.: *The Expression of Algorithms by Charts*. In: Journal of the Association for Computing Machinery, Vol 19, No. 3, July 1972. pp. 517-525.

[8] Bundschuh, P.: *Einführung in die Zahlentheorie*. Springer, Berlin. 1998 [1988].

[9] Cucker, F.; Matamala, M.: *On Digital Nondeterminism*. In: Mathematical Systems Theory, Vol. 29, No. 6, 1996. S.635-647.

[10] Dijkstra, E.W.: *Go To Statement Considered Harmful*. In: Communications of the ACM, Vol. 11, No. 3, 1968. S.147-148.

[11] Ebbinghaus, H.-D.; Flum, J.; Thomas, W.: *Einführung in die mathematische Logik*. Spektrum Akademischer Verlag, Berlin, 1998 [1996].

[12] Ebbinghaus, H.-D.; Flum, J.: *Finite Model Theory*, Springer-Verlag Berlin Heidelberg New York, 1999.

[13] Flum, J: *Modelltheorieskript*. http://home.mathematik.uni-freiburg.de/flum/ss06mod/skript.ps (20.3.2008).

[14] Fournier, H; Koiran, P.: *Lower Bounds Are Not Easier Over The Reals: Inside PH*. LIP Research Report 99-21, Ecole Normale Supérieure de Lyon, 1999. http://citeseer.ist.psu.edu/fournier99lower.html

[15] Goode, J. B.: *Accessible Telephone Directories*. In: Journal Of Symbolic Logic, Vol 59, No.1, 1994. S.92-105

[16] Hemmerling, A.: *On P versus NP for Parameter-Free Programs Over Algebraic Structures*. In: Mathematical Logic Quarterly Vol. 47, No. 1, 2001. S.67 - 92

[17] Ianov, I.: *On The Equivalence And Transformation Of Program Schemes*. In: Communication of the ACM, Vol. 1, Issue 10, 1958. S. 8-12.

[18] Koiran, P.: *A Weak Version of the Blum, Shub & Smale model*. DIMACS Technical Report 94-10, April 1994.

[19] Koiran, P.: *Circuits versus Trees in Algebraic Complexity*. In: Proc. STACS 2000, volume 1770 of Lecture Notes in Computer Science, pages 35-54. Springer-Verlag, 2000.

[20] Michaux, C.: *$P \neq NP$ over the non standard reals implies $P \neq NP$ over* $\mathbb{R}$. In: Theoretical Computer Science, Vol. 133, No. 1, 1994. S. 95-104.

[21] Papadimitriou, C. H.: *Computational Complexity*. Addison-Wesley Publishing Company. 1995 [1994].

[22] Poizat, B.: *Les Petits Cailloux*. Aleas. 1995.

[23] Poizat, B.: *A Course In Model Theory*, An Introduction To Contemporary Mathematical Logic. Springer, New York. 2000.

[24] Prunescu, M.: *Structure With Fast Elimination Of Quantifiers*. In: Journal Of Symbolic Logic, Vol. 71, No. 1, 2006. S.321 - 328

[25] Prunescu, M.: *A model-theoretic proof for $P \neq NP$ over all infinite abelian groups*. In: The Journal of Symbolic Logic, Vol. 67, No. 1, 2002. S. 235-238

[26] Ritter, C.: *Algebraische Komplexität*. Skript, 2007.

[27] Steele, G. L.: *Debunking the „Expensive Procedure Call“ Myth*. In: ACM Annual Conference/Annual Meeting, Proceedings of the 1977 annual conference, 1977. S. 153-162.