

Brandenburgische Technische Universität  
Institut für Informatik und Informations-  
und Medientechnik  
Lehrstuhl Theoretische Informatik



## Diplomarbeit

# Interaktive Beweissysteme in verschiedenen Berechnungsmodellen

Thomas König\*

21. Dezember 2009<sup>†</sup>

Betreuer und Erstgutachter: Prof. Dr. Klaus Meer  
Zweitgutachter: Prof. Dr. Ekkehard Köhler

\*E-Mail: [thomas.koenig@tu-cottbus.de](mailto:thomas.koenig@tu-cottbus.de)

<sup>†</sup>Fassung vom 5. Februar 2010: Korrektur einiger Typographie- und Rechtschreibfehler



# Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Cottbus, 21. Dezember 2009

\_\_\_\_\_  
Unterschrift



# Inhaltsverzeichnis

Abbildungsverzeichnis	vii
Liste der Algorithmen	ix
Einleitung	1
<b>1 Reelle Komplexitätstheorie</b>	<b>5</b>
1.1 Das Blum-Shub-Smale-Modell . . . . .	6
1.2 Elementare Komplexitätsklassen und Entscheidungsprobleme . . . . .	9
1.3 Erste Vollständigkeitsresultate . . . . .	12
1.4 Weitere Komplexitätsklassen . . . . .	21
1.4.1 Alternierende Komplexitätsklassen . . . . .	22
1.4.2 Parallele Komplexitätsklassen . . . . .	23
1.4.3 Beziehungen der alternierenden und parallelen Klassen . . . . .	26
<b>2 Interaktive Beweissysteme</b>	<b>27</b>
2.1 Formale Definition . . . . .	30
2.2 $IP \subseteq PSPACE$ . . . . .	33
2.3 $co-NP \subseteq IP$ . . . . .	35
2.3.1 Arithmetisierung von $\overline{3SAT}$ -Instanzen . . . . .	36
2.3.2 Interaktives Beweissystem für $\overline{3SAT}$ . . . . .	38
2.4 $PSPACE \subseteq IP$ . . . . .	42
2.4.1 $Q3SAT$ ist $PSPACE$ -vollständig . . . . .	42
2.4.2 Arithmetisierung von $Q3SAT$ -Instanzen . . . . .	44
2.4.3 Interaktives Beweissystem für $Q3SAT$ . . . . .	46
2.5 Bibliographische Hinweise . . . . .	51
<b>3 Reelle interaktive Beweissysteme</b>	<b>53</b>
3.1 Formale Definition . . . . .	53
3.2 $BIP_{\mathbb{R}} \subseteq PAR_{\mathbb{R}}$ . . . . .	54
3.3 $PAR_{\mathbb{R}_+} \subseteq BIP_{\mathbb{R}_+}$ . . . . .	55
3.3.1 Ein $PAR_{\mathbb{R}_+}$ -vollständiges Entscheidungsproblem . . . . .	55
3.3.2 Mathematische Hilfsmittel . . . . .	59
3.3.3 Reduktion auf ein diskretes Entscheidungsproblem . . . . .	64
3.4 $BIP_{\mathbb{R}} \not\subseteq IP_{\mathbb{R}_+}$ . . . . .	69
3.4.1 Zwei Ganzzahl-Entscheidungsprobleme . . . . .	70
3.4.2 $(J, INT) \notin PAR_{\mathbb{R}}$ . . . . .	71

## *Inhaltsverzeichnis*

3.4.3	$(\mathcal{P}, \text{PROD}) \in \text{IP}_{\mathbb{R}_+}$ . . . . .	72
3.4.4	$(\mathcal{I}, \text{INT}) \in \text{IP}_{\mathbb{R}_+}$ . . . . .	74
3.4.5	$\text{IP}_{\mathbb{R}_+}$ -Entscheidbarkeit diskreter Entscheidungsprobleme . . .	77
	<b>Zusammenfassung</b>	<b>79</b>
	<b>Literaturverzeichnis</b>	<b>83</b>

# Abbildungsverzeichnis

3.1	$h$ -Dekomposition von $x$ in $\mathbb{R}^2$ , für $h = 3$ und $x = (e, \pi)$ . . . . .	66
-----	---	----



# Liste der Algorithmen

1.1	BSS-Algorithmus zur näherungsweisen Nullstellenbestimmung . . . .	8
2.1	IP-Verifizierer $V$ für $\overline{\text{GI}}$ . . . . .	29
2.2	IP-Verifizierer $V$ für $\overline{\text{3SAT}}$ . . . . .	38
2.3	IP-Verifizierer $V$ für $\text{Q3SAT}$ . . . . .	48
3.1	$\text{BIP}_{\mathbb{R}_+}$ -Protokoll zur $h$ -Dekomposition . . . . .	68
3.2	$\text{BIP}_{\mathbb{R}_+}$ -Verifizierer $V$ für $(\text{HDTRAO}, \text{HDTRAO}_+)$ . . . . .	69
3.3	$\text{P}_{\mathbb{R}_+}$ -Algorithmus für $x \mapsto [x]$ . . . . .	70
3.4	$\text{IP}_{\mathbb{R}_+}$ -Verifizierer $V$ für $(\mathcal{P}, \text{PROD})$ . . . . .	73
3.5	$\text{IP}_{\mathbb{R}_+}$ -Verifizierer $V$ für $(\mathcal{I}, \text{INT})$ . . . . .	75
3.6	BSS-Algorithmus für diskrete Entscheidungsprobleme $L \subseteq \{0, 1\}^*$ . . . .	78



# Einleitung

Jeder, der sich interessiert mit mathematischen Fragestellungen befasst, wird früher oder später die Erfahrung machen, dass es offenbar ungleich schwieriger ist, einen Beweis zu finden als einen bereits gefundenen Beweis zu verstehen.

Um diese oft intuitiv wahrgenommene Diskrepanz besser zu verstehen, wird zunächst eine Definition des Begriffs *Beweis* benötigt. Ein Beweis belegt die Richtigkeit einer Aussage. Er besteht aus einer in geeigneter kalkülhafter Sprache formulierten Schlussfolgerungskette, die es innerhalb des verwendeten formalen logischen Systems erlaubt, die Aussage als wahr zu akzeptieren. Man erwartet dabei allgemein, dass für wahre Aussagen Beweise existieren und dies umgekehrt für falsche Aussagen nicht gilt.

Im Sinne der zuvor erwähnten Diskrepanz ist eine entscheidende Anforderung an Beweise zu stellen, dass sie leicht nachzuvollziehen sind, unabhängig davon, wie schwer es war, sie zu finden. Das Überprüfen von Beweisen soll also effizient durchführbar sein, während dies für das Finden von Beweisen nicht gelten muss. Veranschaulichend lässt sich das Auffinden eines Beweises unter allen denkbaren Schlussfolgerungsketten vergleichen mit der sprichwörtlichen Suche nach der Nadel im Heuhaufen, obschon es sehr einfach ist, eine gefundene Nadel als solche zu identifizieren.

Diese Auffassung eines mathematischen Beweises spiegelt sich in der Definition der Komplexitätsklasse NP wider. Die Klasse NP ist die Menge der Entscheidungsprobleme, deren positive Instanzen sich durch so genannte Zertifikate effizient als solche beweisen lassen, während dies für die negativen Instanzen nicht möglich ist. Ein Zertifikat ist dabei gerade der geeignet notierte Beweis für die Aussage, dass die betrachtete Instanz eine positive ist. Der nichtdeterministische Charakter von NP ist in dieser Definition nicht im zugrunde liegende Maschinenmodell, sondern in der Existenzquantifizierung des benötigten Zertifikats begründet. Die Frage nach der Existenz eines Zertifikats macht dabei die Komplexität der Klasse NP aus.

Nun ist das Veröffentlichen eines gefundenen Beweises z. B. in einer wissenschaftlichen Publikation oder einem Buch nicht der einzig denkbare Weg, den betreffenden Beweis zu kommunizieren. Meist wird derjenige, der versucht den Beweis nachzuvollziehen davon profitieren, Fragen zum Beweis stellen zu dürfen und daraufhin zusätzliche erklärende Antworten zu erhalten. Es ist also naheliegend zu vermuten, dass die Schaffung einer Dialogsituation bei ansonsten gleichem Aufwand das

## Einleitung

Überprüfen von komplizierteren Beweisen, d. h. von Beweisen für kompliziertere Aussagen ermöglicht.

Ein Beispiel für kompliziertere Aussagen sind Instanzen des Nichterfüllbarkeitsproblems aussagenlogischer Formeln. Das Komplement, also das wohlbekanntere Erfüllbarkeitsproblem SAT, erlaubt für den Nachweis der Erfüllbarkeit einer Formel effizient überprüfbare Zertifikate, z. B. in Form einer erfüllenden Variablenbelegung, und ist daher in der Klasse NP enthalten. Für das Nichterfüllbarkeitsproblem hingegen existieren, so wird gemeinhin vermutet<sup>1</sup>, keine die Nichterfüllbarkeit einer gegebenen Formel beweisende und effizient überprüfbare Zertifikate.

Wird nun die Wahrheit einer Aussage in einem Dialog bewiesen, spricht man nicht mehr von einem Zertifikat, sondern von einem *interaktiven Beweis*. Dieser interaktive Beweis belegt wie zuvor das Zertifikat die Wahrheit der betreffenden Aussage. Anstelle der Existenzquantifizierung eines Zertifikats tritt die Existenzquantifizierung eines interaktiven Beweises. Auf diese Weise erhält man die Komplexitätsklasse der *interaktiven Beweissysteme* IP, also die Menge aller Entscheidungsprobleme, für genau deren positive Instanzen effiziente interaktive Beweise existieren.

Ein erstes zentrales Ergebnis zu interaktiven Beweissystemen war das Resultat von Shamir [Sha92], das die Gleichheit der Klassen IP und PSPACE belegt. Dies zeigt, dass effiziente interaktive Beweise wesentlich mächtiger sind<sup>2</sup> als die Zertifikate der Klasse NP. So ist z. B. das oben genannte Nichterfüllbarkeitsproblem aussagenlogischer Formeln effizient mit interaktiven Beweisen entscheidbar.

Shamirs Resultat ist dem Gebiet der klassischen, auf dem Berechnungsmodell der Turingmaschine beruhenden Komplexitätstheorie zuzuordnen. Dennoch fällt auf, dass die verwendete Beweismethode stark algebraisch geprägt ist. Daher liegt die Idee nahe, interaktive Beweise und damit verbundene Komplexitätsklassen in ein algebraisches Berechnungsmodell zu übertragen.

Algebraische Berechnungsmodelle basieren auf algebraischen Strukturen. Sie definieren Maschinen, die über einem algebraischen Ring oder Körper unter Verwendung dazugehöriger Operationen rechnen. In diesem Sinne lassen sich auch klassische Turingmaschinen als algebraische Maschinen auffassen, die über einer endlichen Struktur rechnen, z. B. mit den Operationen  $+$  und  $\cdot$  über dem Restklassenring  $\mathbb{Z}_2$ . Diese sehr allgemein gehaltene Definition geht auf Blum, Shub und Smale zurück [BSS89], die in den entsprechenden Arbeiten jedoch vor allem ein Berechnungsmodell über den reellen Zahlen etablierten. Das so genannte BSS-Modell definiert Maschinen, die über den reellen Zahlen mit Einheitskosten arithmetische Operationen ausführen können.

Ähnlich wie in der klassischen Komplexitätstheorie können auch basierend auf dem BSS-Modell Komplexitätsklassen wie z. B. P und NP definiert werden. Diese

---

<sup>1</sup>gemäß der Vermutung, dass  $NP \neq co-NP$

<sup>2</sup>gemäß der Vermutung, dass die polynomielle Hierarchie nicht kollabiert

werden zur Unterscheidung vom klassischen Fall mit dem Index  $\mathbb{R}$  gekennzeichnet, also  $P_{\mathbb{R}}$  und  $NP_{\mathbb{R}}$ . Die auf diese Weise konstituierte reelle Komplexitätstheorie hält mit dem klassischen Fall vergleichbare Fragestellungen bereit, wie etwa die Frage nach der Gleichheit der Klassen  $P_{\mathbb{R}}$  und  $NP_{\mathbb{R}}$ . Weiter können auch eine reelle polynomielle Hierarchie und darüber hinausgehende Klassen definiert werden, deren Beziehungen untereinander denen im klassischen Fall sehr ähneln.

In diese Hierarchie reeller Komplexitätsklassen lässt sich auch die Klasse der reellen interaktiven Beweissysteme  $IP_{\mathbb{R}}$  einordnen. Zunächst ist jedoch unklar, wie eine reelle Entsprechung des Resultates von Shamir aussehen kann. Da Platzbeschränkungen im reellen Fall nicht zu sinnvollen Komplexitätsklassen führen, wird eine alternative Charakterisierung der Klasse  $PSPACE$  benötigt. Hier bieten sich Schaltkreise polynomieller Tiefe oder polynomiell zeitbeschränkte alternierende Maschinen an. Diese Charakterisierung führt zu den entsprechenden reellen Komplexitätsklassen  $PAR_{\mathbb{R}}$  und  $AP_{\mathbb{R}}$ .

Einen ersten Schritt hin zu der reellen Entsprechung von Shamirs Resultat leisteten Ivanov und de Rougemont [IdR98]. Sie zeigten für eine besondere Form reeller interaktiver Beweise und für eine auf Additionen eingeschränkte Variante des BSS-Modell die Äquivalenz der entsprechenden Komplexitätsklassen. Ein weiteres Resultat war die Separation der auf diese Weise eingeschränkten Klasse reeller interaktiver Beweissysteme von der uneingeschränkten Klasse.

Der Hauptteil der vorliegenden Arbeit besteht darin, die Ergebnisse von Ivanov und de Rougemont umfassend nachzuzeichnen und kurz gehaltene Argumente ausführlich zu ergänzen. Zuvor wird neben einer Darstellung des zugrunde liegenden Resultats von Shamir eine Einführung in die reelle Komplexitätstheorie basierend auf dem BSS-Modell gegeben. Es ergibt sich die folgende Gliederung.

In Kapitel 1 werden das BSS-Modell und darauf basierend nach Einführung eines geeigneten Komplexitätsmaßes erste reelle Komplexitätsklassen definiert. Es folgt der Nachweis der Existenz vollständiger Probleme für die Klasse  $NP_{\mathbb{R}}$ , bevor mit Blick auf die Resultate von Ivanov und de Rougemont weitere Komplexitätsklassen vorgestellt werden. Insbesondere werden als zusätzliches Maschinenmodell die reellen Schaltkreise beschrieben, um nachfolgend die Komplexitätsklasse  $PAR_{\mathbb{R}}$  definieren zu können.

In Kapitel 2 wird Shamirs Resultat  $IP = PSPACE$  vorgestellt. Dazu wird zunächst das Maschinenmodell und der Akzeptierungsbegriff interaktiver Beweissysteme für den klassischen Fall definiert. Nach einigen prinzipiellen Beispielen zur Mächtigkeit interaktiver Beweissysteme folgt zum Beleg der oberen Schranke ein Beweis für  $IP \subseteq PSPACE$ . Zur Vorbereitung des Beweises für die untere Schranke  $PSPACE \subseteq IP$  folgt anschließend unter Einführung der dann später verwendeten Beweistechnik der Nachweis von  $co-NP \subseteq IP$ . Der abschließende Beweis von Shamirs Resultat folgt den in [She92] gegebenen Darstellungen.

## *Einleitung*

In Kapitel 3 werden die Resultate von Ivanov und de Rougemont vorgestellt. Zunächst werden dazu die bezüglich der verwendeten arithmetischen Operationen eingeschränkten Varianten der in Kapitel 1 vorgestellten Komplexitätsklassen, sowie die reellen Varianten interaktiver Beweissysteme definiert. Anschließend wird die grobe Beweisstruktur des ersten Resultats erläutert und dann nach der Vorstellung benötigter mathematischer Hilfsmittel der Äquivalenzbeweis vollständig geführt. Für den nachfolgend geführten Beweis des Separationsresultats wird ein die beiden Klassen trennendes Problem vorgestellt und analysiert.

Die Arbeit schließt mit einer Zusammenfassung der vorgestellten Ergebnisse und einem Ausblick auf weitere interessante Fragestellungen.

In dieser Arbeit sind vergleichende Aussagen die Mächtigkeit der vorgestellten Komplexitätsklassen betreffend, sofern nicht anders vermerkt, stets relativ zur gemeinhin akzeptierten Vermutung zu verstehen, dass die polynomielle Hierarchie nicht kollabiert. Weiter wird in dieser Arbeit das Quadratsymbol  $\square$  durchweg zur Bezeichnung des Endes oder der Abwesenheit eines Beweises verwendet.

# 1 Reelle Komplexitätstheorie

Zahlreiche Berechnungsprobleme aus klassischen Gebieten der Mathematik, wie z. B. das Lösen von linearen Gleichungssystemen oder die näherungsweise Bestimmung von Nullstellen, sind über den reellen Zahlen formuliert. Dennoch bezieht sich die klassische Berechenbarkeits- und Komplexitätstheorie auf diskrete Berechnungsmodelle wie beispielsweise die Turingmaschine.

In diskreten Modellen können viele Problemstellungen, z. B. aus dem Gebiet der Kombinatorik oder der Logik, adäquat modelliert werden. Jedoch scheint ein analoges Vorgehen bei der Untersuchung von Algorithmen über den reellen Zahlen ein ungeeigneter Ansatz zu sein. Eine angemessene Abstraktion ist es vielmehr, die reellen Zahlen im Maschinenmodell als Entitäten zu betrachten, statt sie durch diskrete Repräsentationen zu approximieren.

Die von Blum, Shub und Smale in [BSS89] etablierte reelle Berechenbarkeits- und Komplexitätstheorie basiert auf einem uniformen Maschinenmodell, das Berechnungen über den reellen Zahlen bzw. über beliebigen Ringen ermöglicht. Dies eröffnet analog zur klassischen Komplexitätstheorie einen strukturellen Zugang zur reellen Komplexitätstheorie. So können durch geeignete Ressourcenbeschränkungen und Variationen des Akzeptierungsverhaltens des Maschinenmodells reelle Komplexitätsklassen definiert werden, die denen des klassischen Falls ähneln.

In diesem Kapitel wird eine Einführung in die reelle Komplexitätstheorie basierend auf dem Modell von Blum, Shub und Smale (BSS-Modell) gegeben. Einer Definition des Maschinenmodells im ersten Abschnitt folgen im zweiten Abschnitt die Definitionen der elementaren Komplexitätsklassen  $P_{\mathbb{R}}$  und  $NP_{\mathbb{R}}$ . Für die Klasse  $NP_{\mathbb{R}}$  werden im darauf folgenden Abschnitt zudem vollständige Probleme angegeben und die entsprechenden Vollständigkeitsbeweise vorgestellt.

Im letzten Abschnitt wird analog zum klassischen Fall die reelle polynomielle Hierarchie  $PH_{\mathbb{R}}$  und die darüber hinausgehende alternierende Komplexitätsklasse  $AP_{\mathbb{R}}$  definiert. Daneben werden als alternatives reelles Maschinenmodell die algebraischen Schaltkreise und die entsprechende Komplexitätsklasse  $PAR_{\mathbb{R}}$  vorgestellt. Abschließend werden die Beziehungen dieser Klassen untereinander und im Vergleich zum klassischen Fall erläutert.

## 1.1 Das Blum-Shub-Smale-Modell

Das Berechnungsmodell nach Blum, Shub und Smale [BSS89,BCSS98] ist durch das im Folgenden definierte Maschinenmodell charakterisiert. Dabei ist eine so genannte BSS-Maschine über  $\mathbb{R}$  im Wesentlichen eine Registermaschine, die arithmetische Basisoperationen jeweils mit konstanten Kosten auszuführen im Stande ist und deren Register jeweils eine beliebige reelle Zahl speichern können. Die Eingabe an eine BSS-Maschine ist ebenso wie die Ausgabe der Maschine nach Beendigung der Berechnung ein endliches Tupel reeller Zahlen.

**Definition 1.1.1.** Sei die Menge aller  $k$ -Tupel über  $\mathbb{R}$  definiert als die Menge aller totalen Funktionen von  $\{1, \dots, k\}$  nach  $\mathbb{R}$ , in Zeichen  $\mathbb{R}^k := \{x \mid x: \{1, \dots, k\} \rightarrow \mathbb{R} \wedge x \text{ total}\}$ .

Damit ist für ein  $x \in \mathbb{R}^k$  die Länge von  $x$  gegeben durch  $|x| = k$  und sei für  $1 \leq i \leq k$   $x_i := x(i)$ , sowie  $(x_1, \dots, x_k)$  eine Schreibweise für  $x$ .

Sei ferner die Menge aller endlichen Tupel über  $\mathbb{R}$  definiert durch  $\mathbb{R}^\infty := \bigcup_{k \in \mathbb{N}} \mathbb{R}^k$ .

**Definition 1.1.2.** Für  $\alpha \in \mathbb{R}$ ,  $x \in \mathbb{R}^n$  und  $y \in \mathbb{R}^m$  seien folgende Konkatenationen definiert:  $(\alpha, x) := (\alpha, x_1, \dots, x_n) \in \mathbb{R}^{n+1}$  und  $(x, y) := (x_1, \dots, x_n, y_1, \dots, y_m) \in \mathbb{R}^{n+m}$ .

**Definition 1.1.3.** Eine BSS-Maschine  $M$  über  $\mathbb{R}$  mit der Menge zulässiger Eingaben  $Y \subseteq \mathbb{R}^\infty$ <sup>1</sup> ist eine endliche Menge  $I$  von Instruktionen, die jeweils mit einer Ordnungsnummer aus  $I_\# := \{1, \dots, N\}$  mit  $N = |I|$  durch die Bijektion  $\beta: I_\# \leftrightarrow I$  eindeutig bezeichnet sind.

Eine Konfiguration von  $M$  ist ein Tupel  $(n, d, s, x) \in I_\# \times \mathbb{N} \times \mathbb{N} \times \mathbb{R}^\infty$ . Dabei ist  $n$  die Nummer der als nächstes von  $M$  auszuführenden Instruktion,  $x$  ist der aktuelle Inhalt der Register unter denen  $d$  und  $s$  ein so genanntes Ziel- bzw. ein Quellregister indizieren. Die Startkonfiguration  $K_{\text{start}}$  von  $M$  für eine bestimmte zulässige Eingabe  $y \in Y$  ist gegeben durch  $(1, 1, 1, y)$ . Eine Konfigurationen mit  $n = N$  heißt Endkonfiguration und die Maschine  $M$  stoppt für ein  $K_{\text{ende}} = (N, d, s, x)$  mit Ausgabe  $x$ .

Eine Instruktion  $\beta(n)$  für  $n \in I_\#$  hat einen der drei folgenden Typen und beeinflusst den Übergang der Maschine in eine Folgekonfiguration gemäß den folgenden Definitionen.

**Berechnung** Instruktionen diesen Typs werden wie folgt weiter unterschieden:

compute $[a, b, c, \circ]$  mit  $a, b, c \in \mathbb{N}$ ,  $\circ \in \{+, -, \cdot, \div\}$

diese Instruktion führt eine Berechnung auf den Registern aus. Sei  $x$  der Registersatz vor und  $x'$  der Registersatz nach Ausführung der Instruktion, so ergeben sich  $x'_a := x_b \circ x_c$  und  $\forall i \neq a: x'_i := x_i$ . Die nächste Instruktion ist  $n + 1$  und die Registerindizes  $d$  und  $s$  bleiben unverändert. Es erfolgt also ein Konfigurationsübergang  $(n, d, s, x) \rightarrow (n + 1, d, s, x')$ .

<sup>1</sup>im Folgenden abkürzend BSS-Maschine über  $Y$  genannt

compute $[\circ_d, \circ_s]$  mit  $\circ_d, \circ_s \in \{+1, 1\}$

diese Instruktion lässt den Registersatz unverändert und verändert ausschließlich die Registerindizes  $d$  und  $s$  entsprechend der gewählten Operationen  $\circ_d$  und  $\circ_s$ . Dazu stehen die Inkrementoperation  $+1: \mathbb{N} \rightarrow \mathbb{N}$  mit  $+1(i) = i + 1$  und die Einsoperation  $\mathbf{1}: \mathbb{N} \rightarrow \mathbb{N}$  mit  $\mathbf{1}(i) = 1$  zur Verfügung. Die nächste Instruktion ist  $n + 1$ .

Es erfolgt ein Konfigurationsübergang  $(n, d, s, x) \rightarrow (n + 1, \circ_d(d), \circ_s(s), x)$ .

compute $[a, \alpha]$  mit  $a \in \mathbb{N}, \alpha \in \mathbb{R}$

diese Instruktion überschreibt ein einzelnes Register mit einer vordefinierten Konstante. Es gilt  $x'_a := \alpha$  und  $\forall i \neq a: x'_i := x_i$ . Die nächste Instruktion ist  $n + 1$  und die Registerindizes  $d$  und  $s$  bleiben unverändert.

Es erfolgt also ein Konfigurationsübergang  $(n, d, s, x) \rightarrow (n + 1, d, s, x')$ .

**Verzweigung** Die folgende Instruktion verzweigt den Steuerfluss:

branch $[n']$  mit  $n' \in \{1, \dots, N\}$

Abhängig vom Inhalt des ersten Registers  $x_1$  erfolgt eine Verzweigung des Steuerflusses entweder zur Nachfolgeinstruktion  $n + 1$ , falls  $x_1 < 0$  oder zur Instruktion  $n'$ , falls  $x_1 \geq 0$ . Alle Register sowie die Registerindizes  $d$  und  $s$  bleiben unverändert.

Es erfolgt also ein Konfigurationsübergang  $(n, d, s, x) \rightarrow (n', d, s, x)$ , falls  $x_1 \geq 0$  bzw.  $(n, d, s, x) \rightarrow (n + 1, d, s, x)$ , falls  $x_1 < 0$ .

**Kopieren** Durch die folgende Instruktion erhält die Maschine – hinausgehend über die in den vorigen Instruktionen fest kodierten Registerindizes – Zugriff auf beliebige weitere Register:

copy

diese Instruktion überschreibt das durch  $d$  indizierte Zielregister mit dem Inhalt des durch  $s$  indizierten Quellregisters. Es gilt  $x'_d := x_s$  und  $\forall i \neq d: x'_i := x_i$ . Die nächste Instruktion ist  $n + 1$  und die Registerindizes  $d$  und  $s$  bleiben unverändert.

Es erfolgt also ein Konfigurationsübergang  $(n, d, s, x) \rightarrow (n + 1, d, s, x')$ .

Wenn eine Instruktion die Registerbelegung verändert und dabei ein bisher noch nicht benutztes Register erstmals belegt wird, werden die davor liegenden unbenutzten Register mit 0 initialisiert. So gilt für die copy-Instruktion zusätzlich falls  $d > |x|$ , dass  $\forall |x| < i < d: x'_i := 0$  (analog für die compute-Instruktionen).

Die (endliche) Menge aller in compute-Instruktionen verwendeter Konstanten  $\alpha$  heißt Menge der *Maschinenkonstanten von M*.

*Bemerkung 1.1.4.* Die Definition der compute-Instruktionen unterscheidet sich in [BSS89, BCSS98] geringfügig von der hier angegebenen Variante gemäß [MM97]. Nach Blum, Shub und Smale ist mit jeder compute-Instruktion eine polynomiale oder rationale Abbildung über einer endlichen Zahl von Registern verknüpft,

## 1 Reelle Komplexitätstheorie

welche die Registerbelegung der Folgekonfiguration bestimmt. Derartige Abbildungen können jeweils durch eine konstante Anzahl von Instruktionen gemäß der hier angegebenen Definition simuliert werden.

**Definition 1.1.5.** Eine BSS-Maschine  $M$  über  $Y$  definiert auf natürliche Weise die von  $M$  berechnete partielle Funktion  $\Phi_M: Y \rightarrow \mathbb{R}^\infty$ . Sie ist gegeben durch das Berechnungsergebnis von  $M$  nach Eingabe eines  $y \in Y$ .

Aus praktischen Erwägungen wird festgelegt, dass einer geeigneten Maschine  $M$  vor der eigentlichen Eingabe  $y$  zunächst deren Länge übergeben wird. Ferner hinterlegt die Maschine die Länge des Berechnungsergebnisses im ersten Register. Sei also  $x' = (x'_1, x'_2, \dots)$  die Ausgabe von  $M$  bei Eingabe  $y' = (|y|, y)$ , dann ist die von  $M$  berechnete Funktion  $\Phi_M$  definiert durch  $\Phi_M(y) = x = (x'_2, x'_3, \dots, x'_{x'_1+1})$ .

Entsprechend heißt eine Funktion  $f: Y \rightarrow \mathbb{R}^\infty$  BSS-berechenbar genau dann, wenn es eine BSS-Maschine  $M$  über  $Y$  gibt, die  $f$  berechnet, für die also  $\Phi_M = f$  gilt.

**Beispiel 1.1.6.** Das Newton-Verfahren zur näherungsweise Bestimmung der reellen Nullstellen einer stetig differenzierbaren Funktion  $f: \mathbb{R} \rightarrow \mathbb{R}$  in Abhängigkeit von einem Startwert  $x_0$  besteht in der iterativen Berechnung der Folge

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Die Berechnung wird abgebrochen, wenn der Abstand zweier Folgenglieder eine vorgegebene Schranke  $\varepsilon$  unterschreitet, d. h.  $|x_{n+1} - x_n| < \varepsilon$ . Sei  $f: \mathbb{R} \rightarrow \mathbb{R}$  das Polynom  $f(x) = x^2 + x$  und somit  $f'(x) = 2x + 1$ , dann berechnet der BSS-Algorithmus 1.1 zu einem Startwert  $x_0$  und einer Schranke  $\varepsilon$  eine näherungsweise Nullstelle von  $f$ .

---

### Algorithmus 1.1 : BSS-Algorithmus zur näherungsweise Nullstellenbestimmung

---

**Eingabe :** Startwert  $x$ , Schranke  $\varepsilon$

**Ausgabe :** näherungsweise bestimmte Nullstelle

setze  $x_0 := x$

setze  $n := 0$

**repeat**

    berechne  $x_{n+1} := x_n - ((x_n \cdot x_n + x_n) \div (2 \cdot x_n + 1))$

**if**  $x_{n+1} - x_n \geq 0$  **then**  $\delta := x_{n+1} - x_n$  **else**  $\delta := x_n - x_{n+1}$

    setze  $n := n + 1$

**until**  $\varepsilon \geq \delta \wedge \delta \not\geq \varepsilon$

Ausgabe von  $x_n$

---

## 1.2 Elementare Komplexitätsklassen und Entscheidungsprobleme

Um im BSS-Modell zu elementaren Komplexitätsklassen zu gelangen, wird aus dem Berechenbarkeitsbegriff zunächst die Definition eines Entscheidungsproblems abgeleitet und der Begriff der Laufzeit einer BSS-Maschine definiert.

Danach werden die elementaren reellen Komplexitätsklassen  $P_{\mathbb{R}}$  und  $NP_{\mathbb{R}}$  analog zu ihren Entsprechungen im diskreten Berechnungsmodell eingeführt, sowie zwei wichtige Entscheidungsprobleme der Klasse  $NP_{\mathbb{R}}$  vorgestellt.

**Definition 1.2.1.** Für  $L \subseteq Y \subseteq \mathbb{R}^{\infty}$  und eine BSS-Maschine  $M$  über  $Y$  sei definiert:

1. Die Haltemenge  $\Omega_M$  von  $M$  ist die Menge aller  $y \in Y$ , für die  $\Phi_M(y)$  definiert ist. Die Ausgabemenge von  $M$  ist  $\Phi_M(Y)$ .
2. Die Menge  $L$  heißt *BSS-aufzählbar über  $Y$*  genau dann, wenn  $L$  Ausgabemenge einer BSS-Maschine über  $Y$  ist.
3. Ein Mengenpaar  $(Y, L)$  heißt *Entscheidungsproblem*. Es wird *BSS-entscheidbar* genannt genau dann, wenn es eine BSS-Maschine  $M'$  über  $Y$  gibt, so dass  $\Phi_{M'}$  gleich der charakteristischen Funktion von  $L$  über  $Y$  ist, d. h.  $M'$  entscheidet  $L$  über  $Y$ .

**Definition 1.2.2.** Für eine BSS-Maschine  $M$  über  $Y \subseteq \mathbb{R}^{\infty}$  und ein  $y \in Y$  ist die *Laufzeit von  $M$  auf  $y$*  definiert durch

$$T_M(y) := \begin{cases} \text{Anzahl der von } M \text{ ausgeführten Instruktionen} & \text{falls } y \in \Omega_M \\ \infty & \text{sonst} \end{cases}.$$

**Definition 1.2.3.** Für  $L, L' \subseteq Y \subseteq \mathbb{R}^{\infty}$  sei definiert:

1. Die Klasse  $P_{\mathbb{R}}$  (*deterministic polynomial-time*) umfasst genau die Entscheidungsprobleme  $(Y, L)$ , für die eine BSS-Maschine  $M$  über  $Y$  und Konstanten  $c, k \in \mathbb{N}$  existieren, so dass  $(Y, L)$  von  $M$  entschieden wird und gilt

$$\forall y \in Y: T_M(y) \leq |y|^k + c.$$

2. Die Klasse  $NP_{\mathbb{R}}$  (*nondeterministic polynomial-time*) umfasst genau die Entscheidungsprobleme  $(Y, L)$ , für die ein Entscheidungsproblem  $(Y, L') \in P_{\mathbb{R}}$  und Konstanten  $c, k \in \mathbb{N}$  existieren, so dass gilt

$$\forall y \in Y: y \in L \Leftrightarrow \exists z \in \mathbb{R}^{\infty}: |z| \leq |y|^k + c \wedge (y, z) \in L'.$$

## 1 Reelle Komplexitätstheorie

*Bemerkung 1.2.4.* Im Gegensatz zur klassischen Komplexitätsklasse NP ist im reellen Fall nicht offensichtlich, ob Entscheidungsprobleme aus  $\text{NP}_{\mathbb{R}}$  BSS-entscheidbar sind. Dies hängt mit der überabzählbaren Vielfalt für die Auswahl eines  $z \in \mathbb{R}^{\infty}$  zusammen.

Allerdings kann, wie im Folgenden gezeigt wird, jede Instanz  $y$  eines beliebigen Entscheidungsproblems  $(Y, L) \in \text{NP}_{\mathbb{R}}$  auf ein Polynom  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  vom Grad 4 abgebildet werden, so dass gilt:

$$y \in L \Leftrightarrow \exists x_1, \dots, x_n: f(x_1, \dots, x_n) = 0.$$

Mit einem Verfahren zur Quantorenelimination im reellen Zahlkörper – z. B. nach Tarski [Tar51], vgl. Satz 3.3.16 – kann diese quantifizierte Formel in Logik erster Stufe in eine quantorenfreie Darstellung umgewandelt und anschließend ihre Gültigkeit überprüft werden. Dies gelingt gemäß [Ren92] und analog zum klassischen Fall mit exponentiellem Zeitverlust.

**Definition 1.2.5.** Für ein  $k \in \mathbb{N}$  ist ein *semi-algebraisches System*  $\phi$  über  $\mathbb{R}^k$  eine boolesche Verknüpfung von Gleichungen und Ungleichungen der Form  $p = 0$  oder  $p > 0$ , wobei  $p: \mathbb{R}^k \rightarrow \mathbb{R}$  ein beliebiges  $k$ -variates Polynom mit Koeffizienten aus  $\mathbb{R}$  sei. Erlaubte Verknüpfungen sind  $\wedge$  und  $\vee$ .

Die *Lösungsmenge*  $S_{\phi}$  eines semi-algebraischen Systems  $\phi$  über  $\mathbb{R}^k$  ist genau die Menge von Punkten  $x \in \mathbb{R}^k$ , welche die boolesche Verknüpfung der Gleichungen und Ungleichungen in  $\phi$  erfüllen, also  $S_{\phi} = \{x \in \mathbb{R}^k \mid \phi(x) = \text{WAHR}\}$ .

Ein semi-algebraisches System  $\phi$  heißt *erfüllbar* genau dann, wenn  $S_{\phi} \neq \emptyset$ .

Eine Menge  $S \subseteq \mathbb{R}^k$  heißt *semi-algebraisch* genau dann, wenn ein semi-algebraisches System  $\phi$  existiert, so dass  $S$  Lösungsmenge von  $\phi$  ist, also  $S = S_{\phi}$  gilt.

**Beispiel 1.2.6.** Das semi-algebraische System  $\phi = (x_1^2 + x_2^2 \leq 1) \wedge (x_2 > 0)$  über  $\mathbb{R}^2$  entspricht dem oberhalb der  $x_1$ -Achse liegenden Teil des Einheitskreises. Folgende äquivalente Darstellung von  $\phi$  hat die in obiger Definition geforderte Form.

$$\phi = \left( (x_1^2 + x_2^2 - 1 = 0) \vee (-x_1^2 - x_2^2 + 1 > 0) \right) \wedge (x_2 > 0)$$

**Definition 1.2.7.** Sei  $\mathcal{S}^{(d)}$  die Menge aller semi-algebraischen Systeme über beliebigen  $\mathbb{R}^k$  mit  $k \in \mathbb{N}$ , deren Polynome maximal Grad  $d$  haben. Analog sei  $\mathcal{P}^{(d)}$  die Menge aller Polynome in einer beliebigen Anzahl von Variablen, die maximal Grad  $d$  haben.

1. Sei  $(\mathcal{S}^{(d)}, d\text{-SAS})$  das Entscheidungsproblem, ob ein semi-algebraisches System mit Polynomen höchstens vom Grad  $d$  erfüllbar ist (*semi-algebraic satisfiability*). Die Menge positiver Instanzen ist gegeben durch

$$d\text{-SAS} = \{ \phi \in \mathcal{S}^{(d)} \mid \phi \text{ ist erfüllbar} \}.$$

## 1.2 Elementare Komplexitätsklassen und Entscheidungsprobleme

Die Länge der Kodierung eines semi-algebraischen Systems  $\phi \in d\text{-SAS}$  über  $\mathbb{R}^k$  ist dominiert durch die Kodierungslänge der enthaltenen  $k$ -variaten Polynome. Für ein festes  $d$  haben diese jeweils die Länge  $O(k^d)$ , siehe Punkt 2.

Mit  $m$  als Anzahl der Polynome in  $\phi$  ergibt sich die Länge der Kodierung eines semi-algebraischen Systems für festen Grad  $d$  als  $O(m \cdot k^d)$ .

2. Sei  $(\mathcal{P}^{(d)}, d\text{-FEAS})$  das Entscheidungsproblem, ob ein multivariates Polynom mit Koeffizienten aus  $\mathbb{R}$  und höchstens Grad  $d$  eine reelle Nullstelle hat (*polynomial feasibility*). Hier ist die Menge positiver Instanzen gegeben durch

$$d\text{-FEAS} = \{f \in \mathcal{P}^{(d)} \mid f \text{ hat eine reelle Nullstelle}\}.$$

Jedes Polynom  $f \in \mathcal{P}^{(d)}$  besitzt eine bis auf die Reihenfolge eindeutige Darstellung als Summe von Monomen in folgender Form

$$f(x_1, \dots, x_k) = \sum_{\substack{\alpha \in \{0, \dots, k\}^d \\ \alpha_i \leq \alpha_{i+1}}} a_\alpha x_{\alpha_1} \cdots x_{\alpha_d} \quad \text{wobei } x_0 = 1.$$

Dementsprechend können Polynome aus  $\mathcal{P}^{(d)}$  wie folgt kodiert werden

$$(k, d) \underbrace{(a_\alpha, \alpha_1, \dots, \alpha_d)}_{\text{Einträge für alle } \alpha \text{ mit } a_\alpha \neq 0} \dots$$

Aus der maximalen Anzahl möglicher  $\alpha \in \{0, \dots, k\}^d$  mit  $\alpha_i \leq \alpha_{i+1}$  ergibt sich die Länge der Kodierung eines  $k$ -variaten Polynoms für festen Grad  $d$  als  $O(k^d)$ .

*Bemerkung 1.2.8.* Die gewählte Kodierung der Polynome in  $\mathcal{S}^{(d)}$  und  $\mathcal{P}^{(d)}$  bedingt die Beschränkung des Grades, um die Auswertung der Polynome auf BSS-Maschinen innerhalb einer festen Laufzeitschranke zu ermöglichen.

Wäre der Grad der Polynome nicht beschränkt, ergäbe sich, wegen der konstanten Kodierungslänge z. B. eines Polynoms mit nur einem Monom aber beliebig hohem Grad, keine im Allgemeinen auf BSS-Maschinen realisierbare Laufzeitschranke für die Auswertung von Polynomen.

**Satz 1.2.9.** *Es gilt  $(\mathcal{S}^{(d)}, d\text{-SAS}) \in \text{NP}_{\mathbb{R}}$  sowie  $(\mathcal{P}^{(d)}, d\text{-FEAS}) \in \text{NP}_{\mathbb{R}}$ .*

*Beweis.* Sei  $\phi \in \mathcal{S}^{(d)}$ . Die Polynome in  $\phi$  haben einen durch  $d$  beschränkten Grad. Eine BSS-Maschine kann daher für eine gegebene Variablenbelegung  $z$  in polynomieller Zeit prüfen, ob  $\phi$  durch  $z$  erfüllt wird. Damit gilt gemäß Definition 1.2.3  $(\mathcal{S}^{(d)}, d\text{-SAS}) \in \text{NP}_{\mathbb{R}}$ .

Analog folgt auch  $(\mathcal{P}^{(d)}, d\text{-FEAS}) \in \text{NP}_{\mathbb{R}}$ . □

### 1.3 Erste Vollständigkeitsresultate

Zur Bildung eines Vollständigkeitsbegriffes im BSS-Modell wird zunächst die BSS-Reduzierbarkeit definiert. Es schließen sich Vollständigkeitsbeweise gemäß den Darstellungen in [Cuc93] für die zuvor eingeführten Entscheidungsprobleme der reellen Komplexitätsklasse  $\text{NP}_{\mathbb{R}}$  an.

**Definition 1.3.1.** Für zwei Entscheidungsprobleme  $(Y, L)$  und  $(Y', L')$  sei definiert:

1.  $(Y', L')$  heißt *BSS-reduzierbar in polynomieller Zeit auf  $(Y, L)$*  genau dann, wenn eine BSS-Maschine  $M$  existiert mit
  - a)  $\Phi_M(Y') \subseteq Y$ ,
  - b)  $\forall y' \in Y': y' \in L' \Leftrightarrow \Phi_M(y') \in L$ ,
  - c) und  $M$  arbeitet in *polynomieller Zeit*, d. h.  $\forall y' \in Y': T_M(y') \leq |y'|^k + c$  für geeignete Konstanten  $c, k \in \mathbb{N}$ .

In Zeichen  $(Y', L') \leq_{\mathbb{R}} (Y, L)$ .

2.  $(Y, L)$  heißt  *$\text{NP}_{\mathbb{R}}$ -schwer bezüglich  $\leq_{\mathbb{R}}$*  genau dann, wenn jedes Entscheidungsproblem aus  $\text{NP}_{\mathbb{R}}$  in polynomieller Zeit auf  $(Y, L)$  BSS-reduzierbar ist.  $(Y, L)$  heißt  *$\text{NP}_{\mathbb{R}}$ -vollständig bezüglich  $\leq_{\mathbb{R}}$*  genau dann, wenn zusätzlich  $(Y, L) \in \text{NP}_{\mathbb{R}}$  gilt.

**Satz 1.3.2.**  $(\mathcal{S}^{(d)}, d\text{-SAS})$  ist  $\text{NP}_{\mathbb{R}}$ -vollständig bezüglich  $\leq_{\mathbb{R}}$  für alle  $d \geq 2$ .

*Beweis.* Es genügt zu zeigen, dass  $(\mathcal{S}^{(2)}, 2\text{-SAS})$   $\text{NP}_{\mathbb{R}}$ -vollständig ist. Dafür ist nach Satz 1.2.9 noch zu zeigen, dass  $(\mathcal{S}^{(2)}, 2\text{-SAS})$   $\text{NP}_{\mathbb{R}}$ -schwer ist.

So wird im Folgenden ein Verfahren angegeben, das für ein beliebiges Entscheidungsproblem  $(Y, L) \in \text{NP}_{\mathbb{R}}$  zu jeder Eingabe  $y \in Y$  ein semi-algebraisches System  $\phi \in \mathcal{S}^{(2)}$  effizient bestimmt, welches genau dann erfüllbar ist, wenn  $y \in L$  gilt.

Nach Definition der Klasse  $\text{NP}_{\mathbb{R}}$  (siehe Definition 1.2.3) gibt es zu jedem Entscheidungsproblem  $(Y, L) \in \text{NP}_{\mathbb{R}}$  eine BSS-Maschine  $M$ , welche die charakteristische Funktion folgender Menge  $L'$  in polynomieller Zeit berechnet.

$$L' = \{(y, z) \mid z \text{ ist Beweis polynomieller Länge für } y \in L\}$$

Für die Zugehörigkeit eines  $y \in Y$  zu den positiven Instanzen des Entscheidungsproblems  $(Y, L)$  gilt insbesondere

$$y \in L \Leftrightarrow \exists z \in \mathbb{R}^{\infty} : z \text{ hat polynomielle Länge} \wedge \Phi_M(y, z) = 1.$$

Es genügt nun ein semi-algebraisches System  $\phi_{M(y)} \in \mathcal{S}^{(2)}$  anzugeben, so dass gilt

$$y \in L \Leftrightarrow \exists \tilde{z} \in \mathbb{R}^{\infty} : \phi_{M(y)}(\tilde{z}) = \text{WAHR}.$$

Damit gilt  $y \in L$  genau dann, wenn  $\phi_{M(y)}$  erfüllbar ist und es folgt

$$y \in L \Leftrightarrow \phi_{M(y)} \in 2\text{-SAS}.$$

Das semi-algebraische System  $\phi_{M(y)}$  zu einer gegebenen festen Maschine  $M$  und einer beliebigen Eingabe  $y$  muss nach Definition der BSS-Reduzierbarkeit (siehe Definition 1.3.1) effizient konstruierbar sein und im vorliegenden Fall die folgende Bedingung erfüllen

$$\exists \tilde{z} \in \mathbb{R}^\infty : \phi_{M(y)}(\tilde{z}) = \text{WAHR} \Leftrightarrow \exists z \in \mathbb{R}^\infty : \Phi_M(y, z) = 1.$$

Die Reduktion gelingt, indem der Berechnungsverlauf der Maschine  $M$  geeignet als semi-algebraisches System modelliert wird. Die Modellierungsidee ist hier analog der des Beweises zum Satz von Cook [Coo71] über die klassische NP-Vollständigkeit von SAT.

Da die Länge eines Beweises  $z$  polynomiell in der Länge von  $y$  beschränkt ist, gibt es ohne Einschränkung ein Polynom  $f$ , das die Laufzeit der Maschine  $M$  für eine beliebige Eingabe  $(y, z)$  beschränkt durch  $T_M(y, z) \leq f(|y|)$ . So führt die Maschine bei Eingabe von  $y$  nicht mehr als  $p = f(|y|)$  Instruktionen aus und benutzt dabei keine weiteren als die ersten  $p$  Register. Die Konfiguration von  $M$  hat daher zu jedem der maximal  $p$  Rechenschritte höchstens die Länge  $p + 3$ .

Ein Konfigurationsübergang der Maschine  $M$ , also die Beziehung einer Konfiguration zur nachfolgenden, wird zu jedem Zeitschritt  $t$  mit  $1 \leq t < p$  durch ein semi-algebraisches System  $\phi_{M,t}$  modelliert (klassisch: boolesche Formel). Die Konjunktion aller  $\phi_{M,t}$  zusammen mit semi-algebraischen Systemen  $\phi_{\text{start}_y}$  und  $\phi_{\text{akzept}}$  zur Beschreibung der Start- und einer akzeptierenden Endkonfiguration ergibt das Gesamtsystem  $\phi_{M(y)}$

$$\phi_{M(y)} := \phi_{\text{start}_y} \wedge \bigwedge_{1 \leq t < p} \phi_{M,t} \wedge \phi_{\text{akzept}}.$$

Zur Beschreibung der Konfigurationen werden die folgenden reellen Variablen für jeden Zeitschritt  $1 \leq t \leq p$  in einer entsprechend indizierten Variante eingeführt:

- $N_t$  modelliert die Nummer des als nächsten auszuführenden Befehls,
- $D_t, S_t$  entsprechen dem aktuellen Wert der Quell- und Zielregisterindizes und
- $X_{t,i}$  modelliert für  $1 \leq i \leq p$  den aktuellen Inhalt von Register  $i$ .

Entsprechend der Gesamtzahl dieser Variablen ist  $\phi_{M(y)}$  ein semi-algebraisches System über  $\mathbb{R}^{p^2+3p}$  und die zugehörigen Polynome sind  $(p^2 + 3p)$ -variabel. Zur Vereinfachung der Notation werden die Polynome im Folgenden als Summe ihrer Monome notiert und Gleichungen bzw. Ungleichungen gegebenenfalls umgestellt. So wird z. B. für das Polynom  $f: \mathbb{R}^{p^2+3p} \rightarrow \mathbb{R}$  mit

$$f(N_1, \dots, N_p, D_1, \dots, D_p, S_1, \dots, S_p, X_{1,1}, \dots, X_{p,p}) = X_{a,b} - X_{c,d} \cdot X_{e,f}$$

## 1 Reelle Komplexitätstheorie

bei den Vorkommen in einem semi-algebraischen System statt  $f(\dots) = 0$  zur besseren Verständlichkeit  $X_{a,b} = X_{c,d} \cdot X_{e,f}$  notiert.

Im Folgenden werden für eine Maschine  $M$  und eine Eingabe  $y$  semi-algebraische Systeme angegeben, die zunächst (i) die Startkonfiguration, dann (ii) die möglichen Konfigurationsübergänge und schließlich (iii) eine akzeptierende Endkonfiguration modellieren. Das resultierende Gesamtsystem ist dabei erfüllbar genau dann, wenn ein geeigneter Beweis  $z$  existiert.

### (i) Startkonfiguration

Die Startkonfiguration der Maschine  $M$  wird durch das semi-algebraische System  $\phi_{\text{start}_y}$  modelliert. Darin wird die Konfiguration von  $M$  bei Eingabe von  $y$  und eines beliebigen potentiellen Beweises  $z$  vor Ausführung der ersten Instruktion festgelegt, also zum Zeitpunkt  $t = 1$ . Dies geschieht, indem die Variablen  $N_1, D_1, S_1$  und alle  $X_{1,i}$  wie folgt festgelegt werden:

$$\phi_{\text{start}_y} := \underbrace{N_1=1}_{\text{Instruktion}} \wedge \underbrace{D_1=1}_{\text{Registerindizes}} \wedge \underbrace{S_1=1}_{\text{Länge der Eingabe } (y,z)} \wedge \underbrace{X_{1,1} \geq |y| \wedge X_{1,1} < p}_{\text{unbelegte Register}} \wedge \underbrace{X_{1,2}=y_1 \wedge \dots \wedge X_{1,|y|+1}=y_{|y|}}_{\text{Eingabe } y} \wedge \bigwedge_{|y|+1 < i \leq p} (X_{1,1} \geq i-1 \vee X_{1,i}=0).$$

In  $\phi_{\text{start}_y}$  bleibt der Teil  $z$  der Eingabe  $(y, z)$  an die  $\text{NP}_{\mathbb{R}}$ -Maschine  $M$  unspezifiziert. So überträgt sich die Existenzquantifizierung des Beweises  $z \in \mathbb{R}^{\infty}$  beim Akzeptierungsverhalten von  $M$  auf die Existenzquantifizierung einer Variablenbelegung  $\tilde{z} \in \mathbb{R}^{p^2+3p}$ , welche das System  $\phi_{M(y)}$  erfüllt. Die von der Eingabe  $(y, z)$  nicht belegten Register sind in  $\phi_{\text{start}_y}$  mit 0 initialisiert. Die zu modellierenden deterministischen Konfigurationsübergänge der Maschine  $M$  werden somit für festes  $y$  nur vom Beweis  $z$  abhängig sein.

### (ii) Konfigurationsübergänge

Zu jedem Zeitpunkt  $1 \leq t < p$  ist durch die Konfiguration  $(n, d, s, x)$  der Maschine  $M$  die Nummer  $n \in I_{\#}$  der als nächste auszuführenden Instruktion gegeben oder  $M$  hat mit  $n = N$  bereits eine Endkonfiguration erreicht. Diese Konfiguration ist durch Gleichungen in den Variablen  $N_t, D_t, S_t$  und alle  $X_{t,i}$  modelliert.

Der Übergang in die Konfiguration des Zeitschritts  $t + 1$  hängt deterministisch von der zugehörigen Instruktion  $\beta(n)$  ab. Daher muss das semi-algebraische System  $\phi_{M,t}$  die durch  $N_{t+1}, D_{t+1}, S_{t+1}$  und alle  $X_{t+1,i}$  modellierte Nachfolgekonfiguration entsprechend in Abhängigkeit von  $N_t$  bestimmen. Die Nummer  $N_t$  der im Zeitschritt  $t$  ausgeführten Instruktion ist allerdings unbekannt, da der Ablauf der Berechnung von  $M$  außer von der Eingabe  $y$  auch vom Beweis  $z$  abhängt.

Dieses Problem wird gelöst, indem  $\phi_{M,t}$  als Diskjunktion von semi-algebraischen Teilsystemen dargestellt wird. Diese modellieren unter der Voraussetzung, die

Nummer der nächsten auszuführenden Instruktion wäre ein konkretes  $n$ , den entsprechenden durch  $\beta(n)$  bedingten Konfigurationsübergang.

Da potentiell jede der  $|I_{\#}| = N$  Instruktionen für die als nächste auszuführende in Frage kommt, entspricht diese Lösung der Disjunktion von  $N$  semi-algebraischen Teilsystemen. Der Übersichtlichkeit wegen wird die Menge der Instruktionsnummern nach den Typen der zugehörigen Instruktionen partitioniert und es gilt

$$\phi_{M,t} := \underbrace{\phi_{M,t}^1}_{\text{compute}[a,b,c,\circ]} \vee \underbrace{\phi_{M,t}^2}_{\text{compute}[\circ_d,\circ_s]} \vee \underbrace{\phi_{M,t}^3}_{\text{compute}[a,\alpha]} \vee \underbrace{\phi_{M,t}^4}_{\text{branch}[n']} \vee \underbrace{\phi_{M,t}^5}_{\text{copy}} \vee \underbrace{\phi_{M,t}^6}_{\text{Endkonfiguration}}.$$

Die Instruktionsnummern jeder der verschiedenen Typklassen werden wie folgt um die zur jeweiligen Instruktion gehörigen konkreten Parameter erweitert. Dies ermöglicht eine kompakte Darstellung der einzelnen semi-algebraischen Systeme  $\phi_{M,t}^1, \dots, \phi_{M,t}^5$ . Die Instruktionsnummer  $N$  wird dabei als Kennzeichen einer Endkonfiguration separat durch ein semi-algebraisches System  $\phi_{M,t}^6$  behandelt.

$$\begin{aligned} I_{\#}^1 &:= \{(n, a, b, c, \circ) \mid n \in I_{\#} \wedge n \neq N \wedge \beta(n) = \text{compute}[a, b, c, \circ]\} \\ I_{\#}^2 &:= \{(n, \circ_d, \circ_s) \mid n \in I_{\#} \wedge n \neq N \wedge \beta(n) = \text{compute}[\circ_d, \circ_s]\} \\ I_{\#}^3 &:= \{(n, a, \alpha) \mid n \in I_{\#} \wedge n \neq N \wedge \beta(n) = \text{compute}[a, \alpha]\} \\ I_{\#}^4 &:= \{(n, n') \mid n \in I_{\#} \wedge n \neq N \wedge \beta(n) = \text{branch}[n']\} \\ I_{\#}^5 &:= \{(n) \mid n \in I_{\#} \wedge n \neq N \wedge \beta(n) = \text{copy}\} \end{aligned}$$

Die instruktionsspezifische Modellierung eines Konfigurationsübergangs zu einem Zeitpunkt  $t$  als semi-algebraisches Teilsystem beinhaltet zunächst die Überprüfung der aktuellen Instruktionsnummer. Dann werden abhängig vom Typ der Instruktion gegebenenfalls Veränderungen an einem Register oder an den Registerindizes vorgenommen und zuletzt die Nummer der Folgeinstruktion festgelegt. Die übrigen Komponenten der aktuellen Konfiguration werden unverändert in die Folgekonfiguration übernommen.

Für eine übersichtlichere Notation werden die folgenden Abkürzungen verwendet:

- alle Register bleiben unverändert:  $\phi_{M,t}^X := \bigwedge_{1 \leq i \leq p} (X_{t+1,i} = X_{t,i})$
- alle Register außer Nummer  $j$  bleiben unverändert:  $\phi_{M,t}^{X|j} := \bigwedge_{1 \leq i \neq j \leq p} (X_{t+1,i} = X_{t,i})$
- Ziel- und Quellregisterindex bleiben unverändert:  $\phi_{M,t}^{D,S} := (D_{t+1} = D_t \wedge S_{t+1} = S_t)$

Für die Klasse aller Instruktionen vom Typ  $\text{compute}[a, b, c, \circ]$  ergibt sich das folgende semi-algebraische System. Die Variablen zur Modellierung der Konfiguration von  $M$  zum Zeitpunkt  $t + 1$  ergeben sich hier, abgesehen von den Variablen für das

## 1 Reelle Komplexitätstheorie

Berechnungsergebnis  $X_{t+1,a}$  und die Folgeinstruktion  $N_{t+1}$ , unverändert aus den entsprechenden Variablen für den Zeitpunkt  $t$ .

$$\phi_{M,t}^1 := \bigvee_{(n,a,b,c,o) \in I_{\#}^1} \left[ \overbrace{N_t=n}^{\text{Instruktion}} \wedge \overbrace{X_{t+1,a}=X_{t,b} \circ X_{t,c}}^{\text{Berechnung}} \wedge \overbrace{\phi_{M,t}^{X|a} \wedge \phi_{M,t}^{D,S}}^{\text{keine Änderung}} \wedge \overbrace{N_{t+1}=n+1}^{\text{Folgeinstruktion}} \right]$$

Diese Modellierung berücksichtigt den Sonderfall der Divisionsoperation noch nicht, bei der die rationale Gleichung  $X_{t+1,a} = X_{t,b} \div X_{t,c}$  entsteht. Bei der notwendigen Umformung in eine erfüllbarkeitsäquivalente Polynomgleichung muss der Fall  $X_{t,c} = 0$  berücksichtigt werden und im semi-algebraischen System ergibt sich  $X_{t+1,a} \cdot X_{t,c} - X_{t,b} = 0 \wedge X_{t,c} \neq 0$ .

Das semi-algebraische System für die Instruktionen vom Typ  $\text{compute}[\circ_d, \circ_s]$  ergibt sich ebenfalls unmittelbar aus dem zu modellierenden Konfigurationsübergang. Es sei angemerkt, dass die zur Veränderung der Registerindizes verwendeten Operationen **+1** und **1** trivial als Polynome darstellbar sind.

$$\phi_{M,t}^2 := \bigvee_{(n,\circ_d,\circ_s) \in I_{\#}^2} \left[ \overbrace{N_t=n}^{\text{Instruktion}} \wedge \overbrace{D_{t+1}=\circ_d(D_t) \wedge S_{t+1}=\circ_s(S_t)}^{\text{Indexberechnungen}} \wedge \overbrace{\phi_{M,t}^X}^{\text{keine Änderung}} \wedge \overbrace{N_{t+1}=n+1}^{\text{Folgeinstruktion}} \right]$$

Die Modellierung eines semi-algebraischen Systems für die Menge aller Instruktionen vom Typ  $\text{compute}[a, \alpha]$  ergibt sich analog den vorigen Fällen:

$$\phi_{M,t}^3 := \bigvee_{(n,a,\alpha) \in I_{\#}^3} \left[ \overbrace{N_t=n}^{\text{Instruktion}} \wedge \overbrace{X_{t+1,a}=\alpha}^{\text{Zuweisung}} \wedge \overbrace{\phi_{M,t}^{X|a} \wedge \phi_{M,t}^{D,S}}^{\text{keine Änderung}} \wedge \overbrace{N_{t+1}=n+1}^{\text{Folgeinstruktion}} \right].$$

Die Instruktionen vom Typ  $\text{branch}[n']$  bewirken keine Veränderung der Registerinhalte oder der Registerindizes. Die Auswahl der Nachfolgeinstruktion  $N_{t+1}$  wird in Abhängigkeit vom ersten Register getroffen. Dies wird im semi-algebraischen System durch die Disjunktion zweier exklusiv erfüllbarer Polynomungleichungen erreicht.

$$\phi_{M,t}^4 := \bigvee_{(n,n') \in I_{\#}^4} \left[ \overbrace{N_t=n}^{\text{Instruktion}} \wedge \overbrace{\phi_{M,t}^X \wedge \phi_{M,t}^{D,S}}^{\text{keine Änderung}} \wedge \left( \overbrace{(X_{t,1} \geq 0 \wedge N_{t+1}=n+1)}^{\text{Folgeinstruktion}} \vee \overbrace{(X_{t,1} < 0 \wedge N_{t+1}=n')}^{\text{Verzweigungsinstruktion}} \right) \right]$$

Die semi-algebraischen Systeme für Instruktionen vom Typ  $\text{copy}$  beginnen analog zu denen anderer Instruktionen mit der Überprüfung der Instruktionsnummer  $N_t$ . Um die Indizierung von Ziel- und Quellregister für die Kopieroperation zu

modellieren, muss eine zusätzliche Disjunktion über alle möglichen Wertpaare der Registerindizes eingeführt werden. Da der Wert der Indizes durch die Laufzeit  $p$  der Maschine  $M$  beschränkt ist, genügt es  $p^2$  Wertepaare zu berücksichtigen. Die Kopieroperation wird nur dann mit den nun konstanten Indizes  $d$  und  $s$  durchgeführt, wenn diese gleich den Werten von  $D_t$  und  $S_t$  in der modellierten Konfiguration sind.

$$\phi_{M,t}^5 := \bigvee_{n \in I_{\#}^5} \bigvee_{\substack{1 \leq d \leq p \\ 1 \leq s \leq p}} \left[ \overbrace{N_t = n}^{\text{Instruktion}} \wedge \overbrace{D_t = d \wedge S_t = s}^{\text{Registerauswahl}} \wedge \overbrace{X_{t+1,d} = X_{t,s}}^{\text{Zuweisung}} \wedge \overbrace{\phi_{M,t}^{X,d} \wedge \phi_{M,t}^{D,S}}^{\text{keine Änderung}} \wedge \overbrace{N_{t+1} = n+1}^{\text{Folgeinstruktion}} \right]$$

Die Maschine  $M$  ist durch  $p$  in ihrer Laufzeit beschränkt. Gleichwohl muss  $M$  bei Eingabe  $y$  nicht für jeden Beweis  $z$  diese Laufzeitschranke ausnutzen. Die Maschine wird stattdessen bei Erreichen einer Konfiguration mit  $N$  als Nummer der als nächste auszuführenden Instruktion durch einen Metamechanismus in der Berechnung unterbrochen. Ein solcher Mechanismus steht für ein semi-algebraisches System, das ohne Kenntnis eines Beweises  $z$  erzeugt wurde, nicht zur Verfügung. Stattdessen wird die Laufzeit der modellierten Maschine  $M$  künstlich auf die Laufzeit  $p$  verlängert, indem das semi-algebraische System ohne Veränderung der Konfiguration bei der Instruktionsnummer  $N$  verharrt.

$$\phi_{M,t}^6 := \left[ \overbrace{N_t = N}^{\text{Ende}} \wedge \overbrace{\phi_{M,t}^X \wedge \phi_{M,t}^{D,S}}^{\text{keine Änderung}} \wedge \overbrace{N_{t+1} = N}^{\text{Ende}} \right]$$

Auf diese Weise wird die bei Beendigung der Berechnung zum Zeitpunkt  $t_{\text{ende}} < p$  erreichte Konfiguration bis zum Zeitpunkt  $p$  konstant beibehalten.

(iii) *akzeptierende Endkonfiguration*

Das nachfolgend angegebene semi-algebraische System prüft, ob die modellierte Berechnung der Maschine  $M$  in eine akzeptierende Endkonfiguration führt.

$$\phi_{\text{akzept}} := \overbrace{N_p = N}^{\text{Instruktion}} \wedge \overbrace{X_{p,1} = 1 \wedge X_{p,2} = 1}^{\text{Ergebnis „1“}}$$

Damit ist das Verfahren zur Konstruktion eines semi-algebraisches Systems  $\phi_{M(y)}$  zu einem festen Entscheidungsproblem  $(Y, L) \in \text{NP}_{\mathbb{R}}$  mit der dazugehörigen BSS-Maschine  $M$  für ein beliebiges  $y \in Y$  vollständig angegeben. Der höchste Grad von Polynomen in  $\phi_{M(y)}$  ist 2 und tritt bei der Modellierung der  $\text{compute}[a, b, c, \circ]$ -Instruktionen im Teilsystem  $\phi_{M,t}^1$  auf. Die Reduktion  $(Y, L) \leq_{\mathbb{R}} (\mathcal{S}^{(2)}, 2\text{-SAS})$  ist somit gegeben durch die Abbildung  $y \mapsto \phi_{M(y)}$ .

Um den Beweis abzuschließen, ist (i) die Vollständigkeit und Korrektheit der angegebenen Reduktion sowie (ii) ihre effiziente BSS-Berechenbarkeit zu zeigen.

## 1 Reelle Komplexitätstheorie

### (i) Vollständigkeit und Korrektheit

Es ist für eine Eingabe  $y$  von der Existenz eines geeigneten Beweises  $z$  abhängig, ob die Maschine  $M$  innerhalb der Zeitschranke  $p$  eine akzeptierende Endkonfiguration erreicht, d. h. ob sie mit dem Berechnungsergebnis „1“ stoppt.

Falls ein solches  $z$  existiert, gibt es auch eine Variablenbelegung  $\tilde{z} \in \mathbb{R}^{p^2+3p}$ , die spätestens im Zeitschritt  $p$  einer akzeptierender Konfiguration entspricht und alle semi-algebraischen Teilsysteme für Startkonfiguration und Konfigurationsübergänge sowie für die akzeptierende Endkonfiguration erfüllt. Das semi-algebraische System  $\phi_{M(y)}$  wird daher insgesamt durch  $\tilde{z}$  erfüllt. Die Belegung  $\tilde{z}$  entspricht unmittelbar den während der Berechnung von  $M$  angenommenen Konfigurationen und enthält somit auch den Beweis  $z$  als Teil der modellierten Registerbelegung zum Zeitpunkt  $t = 1$ .

Gibt es umgekehrt keinen geeigneten Beweis  $z$ , mit dem  $M$  innerhalb von  $p$  Schritten in eine akzeptierende Endkonfiguration gelangt, so ist  $\phi_{M(y)}$  unerfüllbar. Denn gäbe es eine  $\phi_{M(y)}$  erfüllende Variablenbelegung  $\tilde{z}$ , würde diese einer Konfigurationsfolge der Maschine  $M$  entsprechen, die in einer akzeptierenden Konfiguration endet. Die semi-algebraischen Teilsysteme zur Modellierung der Konfigurationsübergänge erzwingen aber eine mit der Startkonfiguration beginnende Folge von Konfigurationen, die gemäß den Instruktionen von  $M$  aufeinanderfolgen. Wenn  $\tilde{z}$  also eine akzeptierende Endkonfiguration modelliert, muss  $\tilde{z}$  auch eine Startkonfiguration modellieren. Dies wäre ein Widerspruch zur Annahme, dass kein Beweis  $z$  existiert.

### (ii) effiziente BSS-Berechenbarkeit

Entscheidend für den Nachweis der effizienten BSS-Berechenbarkeit des Reduktionsverfahrens für eine feste Maschine  $M$  ist der Nachweis der polynomiellen Länge der berechneten  $\phi_{M(y)}$  bezüglich der Länge der Eingaben  $y$ .

Die Länge der Kodierung von  $\phi_{M(y)}$  ergibt sich im Wesentlichen aus der Anzahl der Polynome in  $\phi_{M(y)}$ , die in der Kodierung gemäß Definition 1.2.7 Punkt 2 unabhängig von  $y$  eine konstant beschränkte Länge haben. Die Anzahl der Polynome ergibt sich wie folgt:

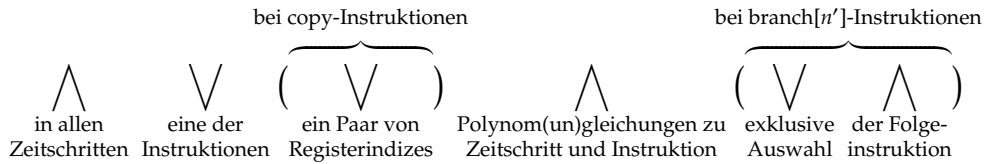
- Das semi-algebraische Teilsystem zur Modellierung der Startkonfiguration benötigt  $O(p)$  Polynome, um alle Register zu initialisieren.
- In  $p - 1$  Zeitschritten geht die modellierte Maschine  $M$  in eine potentiell akzeptierende Endkonfiguration über. Jeder der Zeitschritte besteht aus der Disjunktion einer von  $M$  abhängigen, aber konstanten Anzahl von Teilsystemen, welche jeweils die Ausführung einer Instruktion der Maschine im aktuellen Zeitschritt modellieren. Im ungünstigsten Fall ist jede der Instruktionen eine copy-Instruktion. Für jedes der entsprechenden Teilsysteme werden wegen der Disjunktion über alle möglichen Wertepaare von Registerindizes und der jeweils erfolgenden Veränderung bzw. Übertragung aller Registerinhalte

$O(p^3)$  Polynome benötigt. Insgesamt werden für alle Zeitschritte  $O(p^4)$  Polynome benötigt.

- Das Teilsystem zur Überprüfung, ob die Konfiguration im Zeitschritt  $p$  eine akzeptierende Endkonfiguration ist, benötigt eine konstante Anzahl von Polynomen.

Insgesamt ergibt sich  $O(p^4)$  als Anzahl von Polynomen in  $\phi_{M(y)}$ . Da  $p$  polynomiell von der Länge der Eingabe  $y$  abhängt, ist auch die Länge der Kodierung von  $\phi_{M(y)}$  polynomiell von  $|y|$  abhängig.  $\square$

*Bemerkung 1.3.3.* Die im Beweis zu Satz 1.3.2 angegebene Reduktion  $y \mapsto \phi_{M(y)}$  erzeugt über  $Y$  eine in verschiedener Hinsicht eingeschränkte Teilmenge von 2-SAS. So ist z. B. die Anzahl der Monome in den Polynomen von  $\phi_{M(y)}$  unabhängig von  $y$  konstant beschränkt. Ebenfalls ist die Anzahl der Junktorenwechsel in  $\phi_{M(y)}$  unabhängig von  $y$  konstant beschränkt. Dies ist in abstrakter Notation der Junktorenstruktur von  $\phi_{M(y)}$ , in folgender Darstellung gekürzt auf die Junktorenstruktur von  $\bigwedge_{1 \leq t < p} \phi_{M,t}$ , leicht zu sehen:



Die Reduktion beweist daher die stärkere Aussage, dass  $(\mathcal{S}^{(2)}, 2\text{-SAS}_h)$  für ein geeignet gewähltes  $h \in \mathbb{N}$ , hier z. B.  $h = 4$ ,  $\text{NP}_{\mathbb{R}}$ -vollständig ist. Dabei sei die Anzahl der Junktorenwechsel der semi-algebraischen Systeme aus  $2\text{-SAS}_h$  durch  $h$  beschränkt.

**Satz 1.3.4.**  $(\mathcal{P}^{(d)}, d\text{-FEAS})$  ist  $\text{NP}_{\mathbb{R}}$ -vollständig bezüglich  $\leq_{\mathbb{R}}$  für alle  $d \geq 4$ .

*Beweis.* Es genügt zu zeigen, dass  $(\mathcal{P}^{(4)}, 4\text{-FEAS})$   $\text{NP}_{\mathbb{R}}$ -vollständig ist. Dazu ist es nach Satz 1.2.9 und 1.3.2, nach Bemerkung 1.3.3 und wegen der Transitivität von  $\leq_{\mathbb{R}}$  ausreichend, für ein geeignetes festes  $h$  die Reduzierbarkeit  $(\mathcal{S}^{(2)}, 2\text{-SAS}_h) \leq_{\mathbb{R}} (\mathcal{P}^{(4)}, 4\text{-FEAS})$  nachzuweisen.

Sei  $\phi \in \mathcal{S}^{(2)}$  ein semi-algebraisches System über  $\mathbb{R}^k$  mit einer konstant beschränkten Anzahl von Junktorenwechseln. Die folgenden vier Schritte beschreiben eine Abbildung vom semi-algebraischen System  $\phi$  auf ein Polynom  $f \in \mathcal{P}^{(4)}$ , die in polynomieller Zeit bezüglich der Länge von  $\phi$  BSS-berechenbar ist und für die gilt:

$$\phi \text{ ist erfüllbar} \Leftrightarrow f \text{ hat eine reelle Nullstelle.}$$

### 1. Elimination der Ungleichungen

Das semi-algebraisches System  $\phi \in \mathcal{S}^{(2)}$  kann gemäß Definition 1.2.5 neben Polynomgleichungen der Form  $p(x) = 0$  auch Polynomungleichungen der Form  $p(x) > 0$

## 1 Reelle Komplexitätstheorie

mit  $k$ -variablen Polynomen  $p \in \mathcal{P}^{(2)}$  enthalten. Die Ungleichungen werden ersetzt durch Gleichungen der Form  $q(x, y) = 0$ , wobei  $y$  jeweils eine neue Variable und  $q(x, y) = p(x) \cdot y^2 - 1$  ist. Es entsteht ein semi-algebraisches System  $\phi' \in \mathcal{S}^{(4)}$  über  $\mathbb{R}^{k+O(\text{size}(\phi))}$ , das genau dann erfüllbar ist, wenn auch  $\phi$  erfüllbar ist.

### 2. Elimination der Junktorenstruktur

Anschließend werden in  $\phi'$  mit einem iterativen Verfahren die Disjunktions- bzw. die Konjunktionsketten von Polynomgleichungen  $p_i(x) = 0$  über  $\mathbb{R}^k$  wie folgt schrittweise durch einzelne Polynome ersetzt. Die Länge einer Kette sei durch  $\ell$  und die im Disjunktionsfall benötigten neuen Variablen seien durch  $y_1, \dots, y_\ell$  bezeichnet:

$$\begin{aligned} \bigvee_{1 \leq i \leq \ell} (p_i(x) = 0) &\rightarrow \bigwedge_{1 \leq i \leq \ell} (p_i(x) - y_i = 0) \wedge \left( \prod_{1 \leq i \leq \ell} y_i \right) = 0 \\ \bigwedge_{1 \leq i \leq \ell} (p_i(x) = 0) &\rightarrow \left( \sum_{1 \leq i \leq \ell} p_i(x)^2 \right) = 0. \end{aligned}$$

Es entsteht eine einzelne Polynomgleichung  $p(x) = 0$  über  $\mathbb{R}^{k'+O(\text{size}(\phi))}$ . Das Polynom  $p$  hat polynomiellen Grad in der Länge von  $\phi'$ , da die Zahl der Junktorenwechsel in  $\phi'$  genau wie in  $\phi$  konstant beschränkt ist und nur bei einem Junktorenwechsel eine Verdoppelung des Grades erfolgt. Des Weiteren hat das Polynom  $p$  genau dann eine reelle Nullstelle, wenn das semi-algebraische System  $\phi'$  erfüllbar ist.

### 3. Reduktion auf ein System von Polynomgleichungen über $\mathcal{P}^{(2)}$

Das im vorigen Schritt erhaltene  $k''$ -variante Polynom  $p$  vom Grad  $d$  lässt sich gemäß Definition 1.2.7 Punkt 2 bis auf die Reihenfolge eindeutig als Summe von Monomen darstellen und es entsteht eine Gleichung der Form

$$\sum_{\substack{\alpha \in \{0, \dots, n\}^d \\ \alpha_i \leq \alpha_{i+1}}} a_\alpha x_{\alpha_1} \cdots x_{\alpha_d} = 0.$$

Dabei seien  $x_1, \dots, x_n$  die Variablen des Polynoms und  $x_0 = 1$ , um Monome niedrigeren Grads darstellen zu können. Diese Gleichung wird ersetzt durch die Konjunktion der folgenden linearen Gleichung und quadratischen Gleichungen mit neuen Variablen  $y_\alpha^{(i)}$ .

$$\begin{aligned} \phi''(x, y) := \sum_{\substack{\alpha \in \{0, \dots, k''\}^d \\ \alpha_i \leq \alpha_{i+1}}} a_\alpha y_\alpha^{(1)} = 0 \quad \wedge \\ \bigwedge_{\substack{\alpha \in \{0, \dots, k''\}^d \\ \alpha_i \leq \alpha_{i+1}}} \left( \left( \bigwedge_{1 \leq i < d} y_\alpha^{(i)} = x_{\alpha_i} y_\alpha^{(i+1)} \right) \wedge y_\alpha^{(d)} = x_{\alpha_d} \right) \end{aligned}$$

Da  $p$  polynomielle Länge hat, sind nur polynomiell viele  $a_\alpha \neq 0$ . Da  $p$  außerdem polynomiellen Grad hat, entstehen nur polynomiell viele neue Variablen  $y_\alpha^{(i)}$ .

Insgesamt hat das entstandene semi-algebraische System  $\phi''$  polynomielle Länge. Der Grad ist wie bei dem ursprünglichen System  $\phi$  auf 2 reduziert, jedoch besteht  $\phi''$  ausschließlich aus Konjunktionen von Polynomgleichungen. Das System  $\phi''$  ist darüber hinaus genau dann erfüllbar, wenn das Polynom  $p$  eine reelle Nullstelle besitzt.

4. Reduktion auf eine Polynomgleichung über  $\mathcal{P}^{(4)}$

Zuletzt wird, wie im zweiten Schritt, die Konjunktionskette der quadratischen Gleichungen im semi-algebraischen System  $\phi''$  durch die Summe ihrer Quadrate ersetzt. Es entsteht eine multivariate biquadratische Gleichung  $f = 0$ . Dabei hat das Polynom  $f$  genau dann eine reelle Nullstelle, wenn das semi-algebraische System  $\phi''$  erfüllbar ist. Die Länge von  $f$  ist außerdem polynomiell in der Länge von  $\phi''$ .

Insgesamt gilt für das ursprüngliche semi-algebraische System  $\phi$  und das daraus erhaltene Polynom  $f \in \mathcal{P}^{(4)}$  polynomieller Länge:

$$\phi \text{ ist erfüllbar} \Leftrightarrow f \text{ hat eine reelle Nullstelle.}$$

Die Reduktion  $(\mathcal{S}^{(2)}, 2\text{-SAS}_i) \leq_{\mathbb{R}} (\mathcal{P}^{(4)}, 4\text{-FEAS})$  ist somit durch die in polynomieller Zeit BSS-berechenbare Abbildung  $\phi \mapsto f$  gegeben.  $\square$

## 1.4 Weitere Komplexitätsklassen

Interaktive Beweissysteme beschreiben Komplexitätsklassen, welche die bisher untersuchten Klassen  $P_{\mathbb{R}}$  und  $NP_{\mathbb{R}}$  bezüglich ihrer Mächtigkeit übertreffen. So entsprechen die interaktiven Beweissysteme im klassischen Fall der Komplexitätsklasse PSPACE (siehe Kapitel 2). Auch im reellen Fall ist zu vermuten, dass die interaktiven Beweissysteme in einer Hierarchie reeller Komplexitätsklassen ähnlich einzuordnen sind, wie es der Äquivalenz zu PSPACE im klassischen Fall entspricht.

Zuvor ist allerdings anzumerken, dass bei BSS-Maschinen Platzbeschränkungen nicht zu sinnvollen Komplexitätsklassen führen. Nach [Mic89] oder [dN06] können alle BSS-entscheidbaren Probleme bereits auf konstantem Platz gelöst werden. Jedoch führt die Übertragung der im klassischen Fall impliziten exponentiellen Zeitschranke unter etwas mißbräuchlicher Verwendung der Bezeichnung PSPACE zu nachfolgend definierter platzbeschränkter reeller Komplexitätsklasse  $PSPACE_{\mathbb{R}}$ .

**Definition 1.4.1.** Für eine BSS-Maschine  $M$  über  $Y \subseteq \mathbb{R}^{\infty}$  und ein  $y \in Y$  ist der Platzbedarf von  $M$  auf  $y$  definiert durch

$$S_M(y) := \begin{cases} \text{größter Wert des Quell-/Zielregisterindex} & \text{falls } y \in \Omega_M \\ \infty & \text{sonst} \end{cases}.$$

## 1 Reelle Komplexitätstheorie

**Definition 1.4.2.** Die Klasse  $\text{PSPACE}_{\mathbb{R}}$  (*deterministic polynomial-space*) ist definiert als die Menge genau der Entscheidungsprobleme  $(Y, L)$  mit  $L \subseteq Y \subseteq \mathbb{R}^{\infty}$ , für die eine BSS-Maschine  $M$  über  $Y$  und Konstanten  $c, k \in \mathbb{N}$  existieren, so dass  $(Y, L)$  von  $M$  entschieden wird und gilt

$$\forall y \in Y: T_M(y) \leq 2^{|y|^{k+c}} \wedge S_M(y) \leq |y|^k + c.$$

Im klassischen Fall ist die polynomielle Hierarchie PH in PSPACE enthalten (vgl. [Pap94, Proposition 17.1]) und PSPACE ist äquivalent sowohl zur Klasse PAR der uniformen Schaltkreise polynomieller Tiefe [Bor77] als auch zur Klasse AP der alternierenden polynomiell zeitbeschränkten Turingmaschinen [CKS81].

$$\text{PH} \subseteq \text{PAR} = \text{PSPACE} = \text{AP} \quad (1.1)$$

In den folgenden Abschnitten werden die reellen Entsprechungen  $\text{PH}_{\mathbb{R}}$ ,  $\text{PAR}_{\mathbb{R}}$  und  $\text{AP}_{\mathbb{R}}$  dieser Komplexitätsklassen<sup>2</sup> definiert. Mit dem Ziel, auch im reellen Fall zu einer vergleichbaren Hierarchie von Komplexitätsklassen zu gelangen, werden daran anschließend deren Beziehungen untersucht.

### 1.4.1 Alternierende Komplexitätsklassen

Es folgen Definitionen für die einzelnen Stufen der reellen polynomiellen Hierarchie  $\text{PH}_{\mathbb{R}}$ , sowie für die unbeschränkt alternierende Klasse  $\text{AP}_{\mathbb{R}}$ .

**Definition 1.4.3.** Für  $L, L' \subseteq Y \subseteq \mathbb{R}^{\infty}$  sei definiert:

1. Die Klassen  $\Sigma_k^{\text{P}_{\mathbb{R}}}$  bzw.  $\Pi_k^{\text{P}_{\mathbb{R}}}$  umfassen genau die Entscheidungsprobleme  $(Y, L)$ , für die ein Entscheidungsproblem  $(Y, L') \in \text{P}_{\mathbb{R}}$  und Konstanten  $c, k' \in \mathbb{N}$  existieren, so dass für  $z_1, \dots, z_k \in \mathbb{R}$  mit  $|z_i| \leq |y|^{k'} + c$  gilt

$$\begin{aligned} \text{für } \Sigma_k^{\text{P}_{\mathbb{R}}} \quad \forall y \in Y: \quad y \in L &\Leftrightarrow \exists z_1 \forall z_2 \cdots Q z_k: (y, z_1, \dots, z_k) \in L' \text{ bzw.} \\ \text{für } \Pi_k^{\text{P}_{\mathbb{R}}} \quad \forall y \in Y: \quad y \in L &\Leftrightarrow \forall z_1 \exists z_2 \cdots Q z_k: (y, z_1, \dots, z_k) \in L'. \end{aligned}$$

Dabei seien die  $z_i$  abwechselnd existenz- und allquantifiziert sowie  $Q$  bei geradem bzw. ungeradem  $k$  ein  $\forall$ -Quantor, sonst ein  $\exists$ -Quantor. Weiter sei

$$\Delta_k^{\text{P}_{\mathbb{R}}} = \Sigma_k^{\text{P}_{\mathbb{R}}} \cap \Pi_k^{\text{P}_{\mathbb{R}}}.$$

2. Die Klasse  $\text{PH}_{\mathbb{R}}$  (*polynomial-time hierarchy*) sei definiert als die Vereinigung aller  $\Sigma_k^{\text{P}_{\mathbb{R}}}$ , also

$$\text{PH}_{\mathbb{R}} = \bigcup_{k \in \mathbb{N}} \Sigma_k^{\text{P}_{\mathbb{R}}}.$$

<sup>2</sup>in der Literatur wird die Komplexitätsklasse  $\text{AP}_{\mathbb{R}}$  auch durch  $\text{PAT}_{\mathbb{R}}$  bezeichnet, vgl. [Cuc93]

3. Die Klasse  $AP_{\mathbb{R}}$  (*alternating polynomial-time*) umfasst genau die Entscheidungsprobleme  $(Y, L)$ , für die ein Entscheidungsproblem  $(Y, L') \in P_{\mathbb{R}}$  und Konstanten  $c, k \in \mathbb{N}$  existieren, so dass für  $z_1, z_2, \dots \in \mathbb{R}^{\infty}$  mit  $|z_i| \leq |y|^k + c$  gilt

$$\forall y \in Y: \quad y \in L \Leftrightarrow \exists z_1 \forall z_2 \cdots Q_{z_{|y|^k+c}}: (y, z_1, \dots, z_{|y|^k+c}) \in L'.$$

Dabei seien die  $z_i$  abwechselnd existenz- und allquantifiziert sowie  $Q$  bei geradem  $|y|^k + c$  ein  $\forall$ -Quantor, sonst ein  $\exists$ -Quantor.

**Definition 1.4.4.** Zu einem semi-algebraischen System  $\phi$  über  $\mathbb{R}^k$  sei ein *quantifiziertes semi-algebraisches System*  $F$  definiert durch

$$F = Q_1 x_1 Q_2 x_2 \cdots Q_k x_k \phi(x_1, \dots, x_k).$$

Dabei sei jedes der  $Q_i$  entweder ein Existenz- oder ein Allquantor.

Ein quantifiziertes semi-algebraisches System  $F$  heißt *erfüllbar* genau dann, wenn das zugehörige semi-algebraische System  $\phi$  unter den der Quantifizierung entsprechenden Variablenbelegungen erfüllbar ist.

**Definition 1.4.5.** Sei  $Q^{(d)}$  die Menge aller quantifizierten semi-algebraischen Systeme über beliebigen  $\mathbb{R}^k$  mit  $k \in \mathbb{N}$ , deren Polynome maximal Grad  $d$  haben.

Dann sei  $(Q^{(d)}, d\text{-QSAS})$  das Entscheidungsproblem, ob ein quantifiziertes semi-algebraisches System mit Polynomen höchstens vom Grad  $d$  erfüllbar ist. Die Menge positiver Instanzen ist gegeben durch

$$d\text{-QSAS} = \{F \in Q^{(d)} \mid F \text{ ist erfüllbar}\}.$$

Die Kodierungslänge eines quantifizierten semi-algebraischen Systems  $F \in d\text{-QSAS}$  über  $\mathbb{R}^k$  ist dominiert durch die Länge der Kodierung der enthaltenen  $k$ -variablen Polynome. Mit  $m$  als Anzahl der Polynome in  $F$  ergibt sich die Länge der Kodierung eines quantifizierten semi-algebraischen Systems für ein festes  $d$  zu  $O(m \cdot k^d)$ , siehe auch Definition 1.2.7.

Ähnliche Argumente wie im Beweis der  $NP_{\mathbb{R}}$ -Vollständigkeit von  $(S^{(d)}, d\text{-SAS})$ , siehe Satz 1.3.2, führen schließlich zu folgendem Ergebnis:

**Satz 1.4.6.**  $(Q^{(d)}, d\text{-QSAS})$  ist  $AP_{\mathbb{R}}$ -vollständig bezüglich  $\leq_{\mathbb{R}}$  für alle  $d \geq 2$ . □

## 1.4.2 Parallele Komplexitätsklassen

**Definition 1.4.7.** Ein *algebraischer Schaltkreis*  $C^{(k)}$  über  $\mathbb{R}^k$  ist ein gerichteter azyklischer Graph  $(V, E, <_E, G, O)$  mit Knotenmenge  $V$  und Kantenmenge  $E \subseteq V \times V$ .

Die Knoten des Graphen werden als *Gatter* des Schaltkreises, eingehende Kanten als *Eingänge* und ausgehende Kanten als *Ausgänge* eines Gatters bezeichnet. Die mit

## 1 Reelle Komplexitätstheorie

einem Gatter  $v$  über dessen Eingänge verbundenen Gatter heißen *Eingangsgatter des Gatters  $v$* .

Die Eingänge eines Gatter sind durch  $<_E$  streng geordnet, wobei für zwei Eingänge  $e, e' \in E$  genau dann  $e$  der *linke* und  $e'$  der *rechte Eingang* genannt wird, wenn  $e <_E e'$ .

Die *Gatterfunktion*  $G: V \rightarrow T$  ordnet jedem Gatter einen *Gattertyp* aus der Menge aller Gattertypen  $T = \{\bar{x}_1, \dots, \bar{x}_k, +, -, \cdot, \div, \text{sign}\} \cup \mathbb{R}$  zu.

Weiter unterliegen Kantenmenge  $E$  und Gatterfunktion  $G$  folgenden Einschränkungen bezüglich der Typen und der Zahl der Eingänge je Gatter:

- (i) Es gibt mindestens  $k$  Gatter ohne Eingänge. Von diesen haben genau  $k$  jeweils genau einen der Typen  $\bar{x}_1, \dots, \bar{x}_k$  und werden als *Eingabegatter* bezeichnet.
- (ii) Alle weiteren Gatter ohne Eingänge haben einen der Typen  $c_1, c_2, \dots$  mit  $c_i \in \mathbb{R}$ . Diese Gatter heißen *Konstantengatter*.
- (iii) Die Gatter mit einem Eingang haben den Typ  $\text{sign}$  und heißen *Signumgatter*.
- (iv) Alle übrigen Gatter haben zwei Eingänge und als Typ eine der arithmetischen Operationen  $+, -, \cdot$  oder  $\div$ . Sie werden *Berechnungsgatter* genannt.

Allgemein gibt es in gerichteten azyklischen Graphen mindestens einen Knoten ohne ausgehende Kanten. Im algebraischen Schaltkreis  $C^{(k)}$  heißen diese  $m \geq 1$  Gatter ohne Ausgänge *Ausgabegatter* und sind durch die partielle *Ausgabefunktion*  $O: V \rightarrow \{\bar{y}_1, \dots, \bar{y}_m\}$  eindeutig bezeichnet.

**Definition 1.4.8.** Die von einem algebraischen Schaltkreis  $C^{(k)}$  berechnete Funktion  $f_{C^{(k)}}: \mathbb{R}^k \rightarrow \mathbb{R}^m$  ist wie folgt induktiv definiert. Dazu sei zunächst für jeweils ein  $x \in \mathbb{R}^k$  die Auswertungsfunktion  $\text{eval}_{C^{(k)}}^x: V \rightarrow \mathbb{R}$  definiert

$$\text{eval}_{C^{(k)}}^x v = \begin{cases} x_i & G(v) = \bar{x}_i \\ c_i & G(v) = c_i \\ \text{sign}(\text{eval}_{C^{(k)}}^x v') & G(v) = \text{sign} \wedge (v', v) \in E \\ \text{eval}_{C^{(k)}}^x v' \circ \text{eval}_{C^{(k)}}^x v'' & G(v) \in \{+, -, \cdot, \div\} \wedge \{(v', v), (v'', v)\} \subseteq E \wedge v' <_E v'' \end{cases}$$

Dabei sei die Signumfunktion  $\text{sign}: \mathbb{R} \rightarrow \{0, 1\}$  definiert durch  $\text{sign}(x) = 1$ , falls  $x \geq 0$  und  $\text{sign}(x) = 0$  sonst. Für  $f_{C^{(k)}}$  ergibt sich

$$f_{C^{(k)}}(x_1, \dots, x_k) = (\text{eval}_{C^{(k)}}^x O^{-1}(\bar{y}_1), \dots, \text{eval}_{C^{(k)}}^x O^{-1}(\bar{y}_m)).$$

**Definition 1.4.9.** Für einen algebraischen Schaltkreis  $C^{(k)}$  sei die *Größe*  $s(C^{(k)})$  des Schaltkreises die Anzahl seiner Gatter und die *Tiefe*  $d(C^{(k)})$  des Schaltkreises die Länge des längsten Pfades von einem Eingabe- zu einem Ausgabegatter.

**Definition 1.4.10.** Zu einem algebraischen Schaltkreis  $C^{(k)}$  sei die *kanonische Kodierung von  $C^{(k)}$*  eine Folge von 5-Tupeln der Form  $(v, t, c, v', v'') \in \mathbb{R}^5$ . Jedes der Tupel kodiert eines der Gatter des Schaltkreises wie folgt:  $v \in \{1, \dots, s(C^{(k)})\}$  ist die eindeutige Nummer des Gatters,  $t$  ist der Typ des Gatters und im Falle eines Konstantengatters ist  $c$  die zugehörige Konstante, sonst  $c = 0$ . Weiter sind  $v', v''$  die Nummern des linken bzw. des rechten Eingangsgatters. Dabei sei im Falle eines Eingabe- oder eines Konstantengatters  $v' = v'' = 0$  und im Falle eines Signumgatters  $v'' = 0$ , dessen Eingangsgatter dann durch  $v'$  gegeben ist. Darüber hinaus seien die Eingabegatter  $\bar{x}_1, \dots, \bar{x}_k$  die ersten  $k$  und die Ausgabegatter  $\bar{y}_1, \dots, \bar{y}_m$  die letzten  $m$  in der Nummerierung aller Gatter.

**Definition 1.4.11.** Sei  $\{C^{(k)}\}_{k \in \mathbb{N}}$  eine Familie algebraischer Schaltkreise. Diese heißt  $P_{\mathbb{R}}$ -uniform genau dann, wenn es eine polynomiell zeitbeschränkte BSS-Maschine gibt, welche die folgende Funktion  $g: \mathbb{R}^{\infty} \rightarrow \mathbb{R}^5$  berechnet

$$g(i, 1^k) = i\text{-tes Gatter der kanonischen Kodierung von } C^{(k)}.$$

**Definition 1.4.12.** Die Klasse  $\text{PAR}_{\mathbb{R}}$  (*parallel polynomial-time*) ist definiert als die Menge genau der Entscheidungsprobleme  $(Y, L)$  mit  $L \subseteq Y \subseteq \mathbb{R}^{\infty}$ , für die eine  $P_{\mathbb{R}}$ -uniforme Familie  $\{C^{(k)}\}_{k \in \mathbb{N}}$  von algebraischen Schaltkreisen polynomieller Tiefe derart existiert, dass für alle  $k \in \mathbb{N}$  der Schaltkreis  $C^{(k)}$  die charakteristische Funktion von  $L$  über  $Y$  eingeschränkt auf Eingaben der Länge  $k$  berechnet.

*Bemerkung 1.4.13.* Die Klasse  $\text{PAR}_{\mathbb{R}}$  kann äquivalent definiert werden, als die Menge aller Entscheidungsprobleme, die von einer einfach exponentiellen Anzahl von BSS-Maschinen parallel in polynomieller Zeit gelöst werden können [BCSS98, Kapitel 18]. Die parallel arbeitenden Maschinen können dabei wechselseitig auf ihre Registerinhalte zugreifen.

**Definition 1.4.14.** Sei für das Entscheidungsproblem  $(\mathbb{R}^{\infty}, \text{SCE})$  (*succinct circuit evaluation*) die Menge positiver Instanzen  $\text{SCE}$  gegeben durch die Menge von Tupeln der Form  $(M, x, 1^t, 1^d)$  für die gilt:

- (i)  $M$  ist eine BSS-Maschine die bei Eingabe eines  $i \in \mathbb{N}$  innerhalb von höchstens  $t$  Zeitschritten das  $i$ -te Gatter eines algebraischen Schaltkreises  $C$  ausgibt,
- (ii)  $d(C) \leq d$ ,
- (iii)  $C$  berechnet eine Funktion  $f_C: \mathbb{R}^{|x|} \rightarrow \mathbb{R}$  und
- (iv)  $f_C(x) = 1$ .

Ein Beweis für den folgenden Satz wird in [CB07] geführt.

**Satz 1.4.15.**  $(\mathbb{R}^{\infty}, \text{SCE})$  ist  $\text{PAR}_{\mathbb{R}}$ -vollständig bezüglich  $\leq_{\mathbb{R}}$ . □

### 1.4.3 Beziehungen der alternierenden und parallelen Klassen

In der klassischen Komplexitätstheorie sind, wie bereits angemerkt [Bor77,CKS81], die Ressourcen (i) parallele Zeit, (ii) alternierende Zeit und (iii) Platz unter polynomiellen Beschränkungen gegeneinander austauschbar. Im Folgenden wird eine Reihe von Resultaten zitiert, die für die zuvor definierten reellen Komplexitätsklassen teilweise abweichende Beziehungen belegen.

In [Cuc93] wird mittels eines kurzen Beweises die Simulierbarkeit algebraischer Schaltkreise durch  $\text{PSPACE}_{\mathbb{R}}$ -beschränkte BSS-Maschinen belegt. Zusammen mit dem ebenda angegebenen Beweis der Existenz eines separierenden Entscheidungsproblems folgt

**Satz 1.4.16.** *Es gilt  $\text{PAR}_{\mathbb{R}} \subsetneq \text{PSPACE}_{\mathbb{R}}$ .* □

Ebenfalls in [Cuc93] wird eine allgemeine Reduktion von  $\text{PSPACE}_{\mathbb{R}}$ -beschränkten BSS-Maschinen auf das  $\text{AP}_{\mathbb{R}}$ -vollständige Problem 2-QSAS angegeben und es folgt:

**Satz 1.4.17.** *Es gilt  $\text{PSPACE}_{\mathbb{R}} \subseteq \text{AP}_{\mathbb{R}}$ .* □

Ein bedeutendes algorithmisches Ergebnis über den reellen Zahlen war die nachfolgend angegebene Inklusion (siehe z. B. [Ren92]).

**Satz 1.4.18.** *Es gilt  $\text{PH}_{\mathbb{R}} \subseteq \text{PAR}_{\mathbb{R}}$ .* □

Die zitierten Ergebnisse zusammenfassend ergibt sich folgende Inklusionskette

$$\text{PH}_{\mathbb{R}} \subseteq \text{PAR}_{\mathbb{R}} \subsetneq \text{PSPACE}_{\mathbb{R}} \subseteq \text{AP}_{\mathbb{R}} \tag{1.2}$$

Die strikte Inklusion  $\text{PAR}_{\mathbb{R}} \subsetneq \text{PSPACE}_{\mathbb{R}}$  belegt eine kritische Schranke bezüglich der Mächtigkeit alternierender reeller Komplexitätsklassen. Wenn die Zahl der Quantorenwechsel wie in  $\text{PH}_{\mathbb{R}}$  konstant – obschon nicht universell – beschränkt ist, so können die entsprechenden Entscheidungsprobleme in paralleler polynomieller Zeit gelöst werden. Dies ist bei einer unbeschränkten Anzahl von Quantorenwechseln wie in  $\text{AP}_{\mathbb{R}}$  im Allgemeinen nicht mehr möglich. Es sei angemerkt, dass ein vergleichbares Separationsergebnis für den klassischen Fall nicht bekannt ist (vergleiche Gleichungen 1.1 und 1.2). Weitere Resultate zur Hierarchie alternierender und paralleler Komplexitätsklassen finden sich in [CB07].

## 2 Interaktive Beweissysteme

Viele Komplexitätsklassen sind nicht ausschließlich durch Zeit- oder Platzbeschränkungen eines deterministischen Maschinenmodells charakterisiert. Darüber hinaus werden häufig weitere Anforderungen gestellt, deren Überprüfung die Mächtigkeit des zugrunde liegenden Maschinenmodells übersteigt.

Beispielsweise benötigt die *Verifizierer* genannte deterministische Maschine für ein Entscheidungsproblem  $L$  der Klasse NP für jedes zu akzeptierende Wort  $w$  ein so genanntes *Zertifikat*, das effizient nachprüfbar  $w \in L$  beweist. Die Herkunft dieses Zertifikats liegt der Definition von NP nach außerhalb des zugrunde liegenden deterministischen Maschinenmodells. Eine im Folgenden hilfreiche Interpretation ist, dass das Zertifikat von einem so genannten *Beweiser*, d. h. von einer keinen Beschränkungen unterliegenden Maschine zur Verfügung gestellt wird.

Die Entscheidungsprobleme der Klasse NP sind somit durch einen Monolog des Beweisers charakterisiert, durch den er dem Verifizierer ein effizient überprüfbares Zertifikat zur Verfügung stellt. Hingegen sieht die Klasse der *interaktiven Beweissysteme* IP dem Namen nach einen interaktiven Beweis, also einen Dialog zwischen Verifizierer und Beweiser vor. Dieser verläuft in Runden in denen jeweils eine Nachricht des Verifizierers durch eine Nachricht des Beweisers beantwortet wird. Beide Dialogpartner dürfen die Nachrichten jeweils in Abhängigkeit von den Nachrichten der vorherigen Runden bestimmen. Entscheidend für die Akzeptierung eines Wortes ist hier also die Existenz eines Beweisers, der zusammen mit dem Verifizierer einen für diesen effizient überprüfbaren interaktiven Beweis realisiert.

Vordergründig betrachtet scheinen dem Verifizierer durch die Möglichkeit Antworten des Beweisers in weiteren Runden zu hinterfragen, Mittel an die Hand gegeben, um kompliziertere Beweise zu überprüfen. Dies kann einem deterministischen Verifizierer jedoch nicht gelingen, da der Beweiser dessen deterministisch bestimmte Rückfragen voraussehen und daher den Verlauf des interaktiven Beweises vorherbestimmen kann. Ein deterministischer Verifizierer muss sich deshalb als einem Monolog des Beweisers ausgesetzt betrachten und bleibt somit auf Entscheidungsprobleme der Klasse NP beschränkt.

Eine echte Dialogsituation und damit eine potentiell mächtigere Komplexitätsklasse IP entsteht, wenn der Verifizierer Rückfragen stellen kann, die für den Beweiser unvorhersehbar sind. Der Verifizierer erhält dazu die Möglichkeit auf eine Folge von Zufallsbits zurückzugreifen. Er wird also zu einer probabilistischen Maschine erweitert. Der Beweiser kann die probabilistisch erzeugten Rückfragen nur mit einer

## 2 Interaktive Beweissysteme

gewissen Wahrscheinlichkeit voraussehen. Mit entsprechender Sicherheit kann der Verifizierer die interaktiven Beweise überprüfen.

Die Komplexitätsklasse IP ist dementsprechend definiert als die Menge aller Entscheidungsprobleme  $L$ , für die ein Verifizierer  $V$  existiert, der sich für alle Worte  $w \in L$  von einem Beweiser  $P'$  mit hoher Wahrscheinlichkeit überzeugen lässt,  $w$  zu akzeptieren. Umgekehrt darf für Worte  $w \notin L$  kein Beweiser  $P$  den Verifizierer überzeugen können, mit anderer als geringer Wahrscheinlichkeit zu akzeptieren.

Der Begriff „überzeugen“ bedeutet, dass Beweiser  $P$  und Verifizierer  $V$  abwechselnd und wiederholt Nachrichten austauschen, bis der Verifizierer das Wort  $w$  schließlich akzeptiert oder verwirft, in Zeichen  $\langle P, V \rangle(w) = 1$  oder  $\langle P, V \rangle(w) = 0$ . In formaler Notation ergibt sich

$$\begin{aligned} L \in \text{IP} &\Leftrightarrow \exists \text{ Verifizierer } V \exists \varepsilon > 0: \\ &\forall w \in L \exists \text{ Beweiser } P': \Pr[\langle P', V \rangle(w) = 1] \geq \frac{1}{2} + \varepsilon \quad \wedge \\ &\forall w \notin L \forall \text{ Beweiser } P: \Pr[\langle P, V \rangle(w) = 1] \leq \frac{1}{2} - \varepsilon. \end{aligned}$$

Aus den Definitionen folgt, dass die Komplexitätsklasse NP in der Klasse IP enthalten ist. Hierzu wird ein deterministischer Verifizierer gewählt, der einen einmaligen Rat vom Beweiser erhält. Unklar bleibt zunächst, ob und wie weit IP über NP hinaus geht.

Im nachfolgenden Beispiel wird für das Komplement  $\overline{\text{GI}}$  des Graphisomorphie-Problems (vgl. [SP98]) gezeigt, dass es effiziente interaktive Beweise erlaubt und somit in IP enthalten ist. Gleichwohl wird allgemein vermutet, dass  $\overline{\text{GI}}$  nicht in NP enthalten ist. Somit ist bereits aus diesem Beispiel zu folgern, dass die Komplexitätsklasse IP vermutlich echt über die Klasse NP hinausgeht.

**Beispiel 2.0.1.** Zwei Graphen  $G_1 = (V_1, E_1)$  und  $G_2 = (V_2, E_2)$  heißen *isomorph* genau dann, wenn eine Bijektion  $f: V_1 \leftrightarrow V_2$  derart existiert, dass für alle  $x, y \in V_1$  gilt

$$(x, y) \in E_1 \Leftrightarrow (f(x), f(y)) \in E_2.$$

Offensichtlich ist das Graphisomorphie-Problem GI in der Komplexitätsklasse NP enthalten. Ein geeignetes Zertifikat zum Nachweis der Isomorphie zweier Graphen wäre der Isomorphismus  $f$  selbst.

Umgekehrt ist nicht offensichtlich wie ein Zertifikat für die Nicht-Isomorphie zweier Graphen  $G_1, G_2$  aussehen könnte, da es dem Nachweis entspräche, dass alle Bijektionen  $f: V_1 \leftrightarrow V_2$  die Isomorphieeigenschaft nicht erfüllen. Allerdings belegt der in Algorithmus 2.1 angegebene Verifizierer  $V$ , dass für das Entscheidungsproblem  $\overline{\text{GI}}$  effiziente interaktive Beweise existieren.

Der zufällige, zu  $G_i$  isomorphe Graph  $H$  kann effizient durch eine zufällige Permutation und anschließende Umbenennung der Knoten von  $G_i$  bestimmt werden.

---

**Algorithmus 2.1** : IP-Verifizierer  $V$  für  $\overline{\text{GI}}$ 

---

**Eingabe** : zwei Graphen  $G_1$  und  $G_2$

**Ausgabe** : akzeptiere, falls  $G_1$  nicht isomorph zu  $G_2$ , sonst verwerfe

**wiederhole** *zweimal*

wähle Zufallszahl  $i \in \{1, 2\}$

erzeuge zufällig einen zu  $G_i$  isomorphen Graphen  $H$

sende  $H$  an den Beweiser

empfange  $j \in \{1, 2\}$  vom Beweiser

prüfe, ob  $i = j$ , sonst verwerfe

akzeptiere

---

Angenommen die Graphen  $G_1$  und  $G_2$  sind nicht isomorph, d. h.  $(G_1, G_2) \in \overline{\text{GI}}$ . Dann kann ein Beweiser  $P$  z. B. durch Enumeration aller Bijektionen das eindeutige  $j$  bestimmen, so dass  $H$  isomorph zu  $G_j$  ist. Dieses  $j$  ist gleich der zuvor gewählten Zufallszahl  $i$ , da  $H$  isomorph zu  $G_i$  gewählt wurde. Somit folgt

$$(G_1, G_2) \in \overline{\text{GI}} \Rightarrow \exists \text{Beweiser } P: \Pr[\langle P, V \rangle(G_1, G_2) = 1] = 1.$$

Seien umgekehrt die beiden Graphen isomorph, dann ist der Graph  $H$  isomorph zu  $G_1$  und zu  $G_2$  unabhängig von der Wahl der Zufallszahl  $i$ . Es gibt für einen Beweiser deshalb keine Möglichkeit zu entscheiden, ob der Verifizierer  $H$  aus  $G_1$  oder aus  $G_2$  erzeugt hat. Entsprechend wird das vom Beweiser bestimmte  $j$  mit einer Wahrscheinlichkeit von  $\frac{1}{2}$  mit dem vom Verifizierer gewählten  $i$  übereinstimmen. Nach zwei Wiederholungen ergibt sich eine Wahrscheinlichkeit von  $\frac{1}{4}$  für das fälschliche akzeptieren zweier isomorpher Graphen und es gilt

$$(G_1, G_2) \notin \overline{\text{GI}} \Rightarrow \forall \text{Beweiser } P: \Pr[\langle P, V \rangle(G_1, G_2) = 1] = \frac{1}{4}.$$

Somit folgt die Behauptung und es gilt  $\overline{\text{GI}} \in \text{IP}$ . □

Tatsächlich wird in den folgenden Abschnitten gezeigt, dass die Komplexitätsklasse IP wesentlich über NP hinausgeht<sup>1</sup>. Dazu wird in Abschnitt 2.3 die Technik der Arithmetisierung vorgestellt mit der Lund et al. [LFKN92] zunächst der Beweis gelang, dass die Klasse IP die polynomielle Hierarchie PH enthält. Darauf basierend gelang Shamir [Sha92] der Beweis der Inklusion PSPACE  $\subseteq$  IP.

Zur Demonstration der Beweistechnik wird zunächst co-NP  $\subseteq$  IP gezeigt, ehe der Beweis ergänzt und in Abschnitt 2.4 entsprechend [She92] zu einem Nachweis für PSPACE  $\subseteq$  IP vervollständigt wird.

---

<sup>1</sup>unter der Annahme, dass die polynomielle Hierarchie nicht kollabiert

Doch zuvor folgt im nächsten Abschnitt eine formale Definition dessen, was unter einem interaktiven Beweissystem zu verstehen ist. Aufbauend darauf wird in Abschnitt 2.2 die Inklusion der Komplexitätsklasse IP in PSPACE bewiesen. So ergibt sich als Gesamtergebnis die vollständige Charakterisierung der Klasse IP durch die deterministische Komplexitätsklasse PSPACE.

## 2.1 Formale Definition

Die folgenden Definitionen abstrahieren den Begriff des interaktiven Beweissystems von konkreten Berechnungsmodellen. Dies geschieht mit dem Ziel, eine einheitliche Grundlage zur Definition von Komplexitätsklassen sowohl für die klassischen, wie auch für die in Kapitel 3 untersuchten reellen interaktiven Beweissysteme zu erhalten. Ferner beruht der in Abschnitt 2.2 angegebene Beweis für  $IP \subseteq PSPACE$  sowie der entsprechende Beweis für  $BIP_{\mathbb{R}} \subseteq PAR_{\mathbb{R}}$  in Abschnitt 3.2 auf den hier getroffenen Definitionen.

**Definition 2.1.1.** Ein *Verifizierer*  $V$  über  $Y$  ist eine Familie  $\{V_i\}_{i \in \mathbb{N}}$  von Funktionen der Form  $V_i: \{0, 1\}^* \times Y \times \mathcal{P}^i \rightarrow \{\text{STOPP}, \text{WEITER}\} \times \mathcal{V}$ . Dabei ist  $\mathcal{V}$  die Menge aller möglichen *Nachrichten vom Verifizierer an den Beweiser* und umgekehrt  $\mathcal{P}$  die Menge aller möglichen *Nachrichten vom Beweiser an den Verifizierer*.

Ein *Beweiser*  $P$  über  $Y$  ist eine Familie  $\{P_i\}_{i \geq 1}$  von Funktionen der Form  $P_i: Y \times \mathcal{V}^i \rightarrow \mathcal{P}$ .

Für einen Beweiser  $P$  über  $Y$  und einen Verifizierer  $V$  über  $Y$  ist ein *interaktives Beweissystem über  $Y$*  die im Folgenden für beliebige *Zufallsfolgen*  $\sigma \in \{0, 1\}^*$  induktiv definierte Funktion  $\langle P, V \rangle_{\sigma}: Y \rightarrow \mathcal{V}$  mit

$$\begin{aligned} (\omega_i, v_i) &:= V_i(\sigma, y, p_1, \dots, p_i) \\ p_i &:= P_i(y, v_0, \dots, v_{i-1}) \\ r &:= \min_i [\omega_i = \text{STOPP}] \\ \langle P, V \rangle_{\sigma}(y) &:= \begin{cases} v_r & \text{falls } r < \infty \\ \text{undefiniert} & \text{sonst} \end{cases} \end{aligned}$$

Das bei der Auswertung von  $\langle P, V \rangle_{\sigma}(y)$  bestimmte  $r$  gibt die vom interaktiven Beweissystem benötigte *Anzahl von Runden*  $R_{\langle P, V \rangle_{\sigma}}(y) := r$  an. Weiter wird die bei der Auswertung entstandene Folge  $(v_0, p_1, v_1, p_2, \dots, v_{r-1}, p_r, v_r) \in (\mathcal{V} \times \mathcal{P})^r \times \mathcal{V}$  *interaktiver Beweis für  $y$  bezüglich  $\sigma$* , in Zeichen  $P_{\langle P, V \rangle_{\sigma}}(y)$ , genannt.

**Definition 2.1.2.** Ein interaktives Beweissystem  $\langle P, V \rangle_{\sigma}$  heißt *berechenbar über  $Y$  bezüglich eines Berechenbarkeitsmodells* genau dann, wenn alle Funktionen  $V_i$  des Verifizierers  $V = \{V_i\}_{i \in \mathbb{N}}$  bezüglich des Berechenbarkeitsmodells *uniform berechenbar über  $Y$*  sind. Dies gilt genau dann, wenn die Funktion  $\Phi_V: \mathbb{N} \times \{0, 1\}^* \times Y \times \mathcal{P}^{\infty} \rightarrow \mathcal{V}$  mit  $\Phi_V(i, \sigma, y, p_1, p_2, \dots) = V_i(\sigma, y, p_1, \dots, p_i)$  bezüglich des Berechenbarkeitsmodells berechenbar über  $Y$  ist.

*Bemerkung 2.1.3.* Eine äquivalente Definition der Berechenbarkeit eines interaktiven Beweissystems ergibt sich aus einem wie folgt modifizierten Maschinenmodell. Eine so genannte interaktive Maschine kann nach der Berechnung von  $v_0$  in einen besonderen Kommunikationszustand wechseln. Nach dem Austausch der beiden Nachrichten  $v_0$  und  $p_1$  zwischen der Maschine und dem Beweiser ist die erste Runde des interaktiven Beweises abgeschlossen. Die interaktive Maschine kann anschließend mit der Berechnung von  $v_1$  fortfahren, bis sie gegebenenfalls nach  $r$  Runden mit der Ausgabe von  $v_r$  terminiert.

**Definition 2.1.4.** Ein interaktives Beweissystem  $\langle P, V \rangle_\sigma$  heißt *effizient berechenbar über  $Y$  bezüglich eines Berechenbarkeitsmodells* genau dann, wenn Konstanten  $c, k \in \mathbb{N}$  existieren, so dass folgende Bedingungen erfüllt sind.

- (i)  $\langle P, V \rangle_\sigma$  ist berechenbar über  $Y$  vermöge einer Maschine  $M$  mit  $\Phi_M = \Phi_V$
- (ii)  $\forall y \in Y: R_{\langle P, V \rangle_\sigma}(y) \leq |y|^k + c$
- (iii)  $\forall y \in Y \quad \forall i \leq R_{\langle P, V \rangle_\sigma}(y): T_M(i, \sigma, y, p_1, p_2, \dots) \leq |y|^k + c$

Gemäß obiger Definition ist die Effizienz eines interaktiven Beweissystems charakterisiert durch (i) dessen Berechenbarkeit sowie die polynomielle Beschränkung (ii) seiner Rundenzahl und (iii) der Laufzeit der zugehörigen Maschine im jeweiligen Berechenbarkeitsmodell.

Die im Folgenden definierte Komplexitätsklasse DIP der deterministischen interaktiven Beweissysteme basiert auf einer Erweiterung des Zertifikatbegriffs der Klasse NP. Die Akzeptierungsentscheidungen für Sprachen aus DIP hängen deterministisch von der Existenz eines Beweisers und somit von der Existenz eines interaktiven Beweises ab, der die Sprachzugehörigkeit eines Wortes belegt. Der nachfolgend bewiesene Satz zeigt, dass die so definierte Klasse nicht über NP hinaus geht.

**Definition 2.1.5.** Die Klasse DIP (*deterministic interactive polynomial-time*) ist für  $\mathcal{P} = \mathcal{V} = \{0, 1\}^*$  definiert als die Menge genau der Entscheidungsprobleme  $L \subseteq \{0, 1\}^*$ , für die ein Verifizierer  $V$  über  $\{0, 1\}^*$  existiert, so dass für die leere Zufallsfolge  $\lambda$  folgende Bedingungen erfüllt sind.

- (i)  $\langle P, V \rangle_\lambda$  ist effizient Turing-berechenbar über  $\{0, 1\}^*$
- (ii)  $\forall w \in L \quad \exists \text{Beweiser } P: \langle P, V \rangle_\lambda(w) = 1$
- (iii)  $\forall w \in \{0, 1\}^* \setminus L \quad \forall \text{Beweiser } P: \langle P, V \rangle_\lambda(w) = 0$

Die Eingaben  $w \in \{0, 1\}^*$ , für die gemäß (ii)  $w \in L$  gilt, werden als *vom Verifizierer  $V$  akzeptiert* bezeichnet. Die übrigen Eingaben, für die gemäß (iii)  $w \in \{0, 1\}^* \setminus L$  gilt, werden dagegen als *vom Verifizierer  $V$  verworfen* bezeichnet. Der Verifizierer  $V$  entscheidet  $L$  über  $\{0, 1\}^*$ .

**Satz 2.1.6.** *Es gilt  $\text{DIP} = \text{NP}$ .*

## 2 Interaktive Beweissysteme

*Beweis.* Offensichtlich gilt  $NP \subseteq DIP$ . Der umgekehrte Fall  $DIP \subseteq NP$  folgt aus der Tatsache, dass genau für die Worte  $w \in L$  ein Beweiser  $P$  und somit ein interaktiver Beweis  $P_{\langle P, V \rangle_\lambda}(w)$  existiert, der  $w \in L$  beweist. Ist ein solcher interaktiver Beweis  $z = (v_0, p_1, \dots, v_{r-1}, p_r, v_r)$  mit  $v_r = 1$  als Zertifikat gegeben, kann mit Kenntnis des zugehörigen Verifizierers  $V$  effizient deterministisch geprüft werden, ob die  $v_i$  den Nachrichten des Verifizierers  $V$  entsprechen, falls ein Beweiser  $P'$  die Antworten  $p_i$  senden würde. Der interaktive Beweis  $z$  wäre somit ein Zertifikat für die Existenz eines Beweisers  $P'$  mit  $\langle P', V \rangle(w) = 1$ .  $\square$

Mit dem Ziel eine potentiell mächtigere Komplexitätsklasse als  $NP$  zu erhalten, wird dem Verifizierer die Möglichkeit gegeben, den Verlauf des interaktiven Beweises von Zufallsentscheidungen abhängig zu machen. Die Akzeptierungsentscheidung ist somit nicht mehr von der Existenz eines einzelnen interaktiven Beweises sondern von der Existenz eines Beweisers abhängig, der für beliebige Zufallsfolgen geeignete interaktive Beweise führen kann.

**Definition 2.1.7.** Die Klasse  $IP$  (*interactive polynomial-time*) ist für  $\mathcal{P} = \mathcal{V} = \{0, 1\}^*$  definiert als die Menge genau der Entscheidungsprobleme  $L \subseteq \{0, 1\}^*$ , für die ein Verifizierer  $V$  über  $\{0, 1\}^*$  und eine Konstante  $\varepsilon > 0$  existieren, so dass folgende Bedingungen erfüllt sind.

- (i)  $\forall \sigma \in \{0, 1\}^* : \langle P, V \rangle_\sigma$  ist effizient Turing-berechenbar über  $\{0, 1\}^*$
- (ii)  $\forall w \in L \quad \exists \text{Beweiser } P : \Pr_\sigma \left[ \langle P, V \rangle_\sigma(w) = 1 \right] \geq \frac{1}{2} + \varepsilon$
- (iii)  $\forall w \in \{0, 1\}^* \setminus L \quad \forall \text{Beweiser } P : \Pr_\sigma \left[ \langle P, V \rangle_\sigma(w) = 0 \right] \geq \frac{1}{2} + \varepsilon$

Die Eingaben  $w \in \{0, 1\}^*$ , für die gemäß (ii)  $w \in L$  gilt, werden als *vom Verifizierer  $V$  akzeptiert* bezeichnet. Die übrigen Eingaben, für die gemäß (iii)  $w \in \{0, 1\}^* \setminus L$  gilt, werden dagegen als *vom Verifizierer  $V$  verworfen* bezeichnet. Der Verifizierer  $V$  entscheidet  $L$  über  $\{0, 1\}^*$ .

Somit gehört ein Entscheidungsproblem per obiger Definition genau dann zur Komplexitätsklasse  $IP$ , wenn es unter folgendem probabilistischen Akzeptierungsbegriff effiziente interaktive Beweise erlaubt:

- Gemäß Punkt (ii) der Definition muss für jedes Wort  $w \in L$  ein Beweiser  $P$  derart existieren, dass die meisten interaktiven Beweise  $P_{\langle P, V \rangle_\sigma}(w)$  bezüglich beliebiger Zufallsfolgen  $\sigma$  einen Beleg für  $w \in L$  liefern. Diese Eigenschaft wird *Vollständigkeit* genannt.
- Ferner wird gemäß Punkt (iii) der Definition *Korrektheit* gefordert. Dies bedeutet im Fall  $w \notin L$ , dass die meisten mit einem beliebigen Beweiser geführten interaktiven Beweise keinen Beleg für  $w \in L$  liefern.

Nach [FGM<sup>+</sup>89] bzw. [GMS87] ändert sich die Komplexitätsklasse IP überraschenderweise nicht, wenn in Abwandlung der obigen Definition *perfekte Vollständigkeit* gefordert wird. Dies bedeutet, dass zu jedem Entscheidungsproblem  $L \in \text{IP}$  ein Verifizierer  $V'$  existiert, der die Bedingung  $\forall w \in L \exists \text{Beweiser } P: \Pr_\sigma[\langle P, V' \rangle_\sigma(w)=1] = 1$  erfüllt. In den folgenden Beweisen für  $\text{co-NP} \subseteq \text{IP}$ ,  $\text{PSPACE} \subseteq \text{IP}$  sowie in Kapitel 3 für  $\text{PAR}_{\mathbb{R}} \subseteq \text{BIP}_{\mathbb{R}}$  und  $\text{PAR}_{\mathbb{R}_+} \subseteq \text{BIP}_{\mathbb{R}_+}$  werden Verifizierer angegeben, die bereits perfekte Vollständigkeit aufweisen.

## 2.2 IP $\subseteq$ PSPACE

Um  $\text{IP} \subseteq \text{PSPACE}$  zu beweisen, genügt es zu zeigen, dass für jedes Entscheidungsproblem  $L \in \text{IP}$  die maximale Wahrscheinlichkeit mit der ein zugehöriges interaktives Beweissystem  $\langle P, V \rangle(w)$  unter Beteiligung eines optimalen Beweisers  $P$  ein Wort  $w \in \{0, 1\}^*$  akzeptiert, von einer PSPACE-Maschine bestimmt werden kann. Überschreitet diese maximale Wahrscheinlichkeit die Schranke  $\frac{1}{2} + \varepsilon$ , so gilt  $w \in L$ .

Es ist dabei nicht nötig den optimalen Beweiser  $P$  explizit anzugeben. Stattdessen wird die maximale Wahrscheinlichkeit durch eine Enumeration aller möglichen Beweiser und der Simulation ihrer Interaktion mit dem Verifizierer  $V$  berechnet.

Im Folgenden wird ein rekursives Verfahren angegeben, das für jede Runde unter Berücksichtigung der bisherigen Nachrichten  $v_0, \dots, v_i$  des Verifizierers und über alle  $\sigma$  die für den weiteren Verlauf des interaktiven Beweises optimale Antwort  $p_{i+1}$  bestimmt. Das Berechnungsergebnis ist dabei die Anzahl der  $\sigma$ , für die das interaktive Beweissystem  $\langle P, V \rangle_\sigma(w)$  das Wort  $w$  schließlich als zu  $L$  gehörig akzeptiert. Da die Zahl der Zufallsbits, die der Verifizierer  $V$  verarbeiten kann durch dessen Laufzeitpolynom  $|w|^k + c$  beschränkt ist, genügt es  $\sigma \in \{0, 1\}^{|w|^k + c}$  zu wählen.

**Definition 2.2.1.** Sei für  $m \leq n$  die *Präfixrelation*  $\sqsubseteq$  definiert durch

$$(a_1, \dots, a_m) \sqsubseteq (b_1, \dots, b_n) \Leftrightarrow \forall i \leq m: a_i = b_i.$$

Nun entspricht  $(v_0, p_1, \dots, v_{i-1}, p_i) \sqsubseteq P_{\langle P, V \rangle_\sigma}(w) \wedge \langle P, V \rangle_\sigma(w) = 1$  der Aussage, dass der von  $P$  und  $V$  geführte interaktive Beweis für  $w$  bezüglich  $\sigma$  mit der Nachrichtenfolge  $(v_0, p_1, \dots, v_{i-1}, p_i)$  beginnt und das Wort  $w$  abschließend akzeptiert wird.

**Definition 2.2.2.** Seien für  $L \in \text{IP}$  und den dazugehörigen Verifizierer  $V$  die Funktionen  $Q_i: \{0, 1\}^* \times (\mathcal{V} \times \mathcal{P})^i \rightarrow \mathbb{N}$  und  $W_i: \{0, 1\}^* \times (\mathcal{V} \times \mathcal{P})^i \times \mathcal{V} \rightarrow \mathbb{N}$  definiert:

$$\begin{aligned} Q_i(w, v_0, p_1, \dots, v_{i-1}, p_i) &:= \\ &\max_P \left| \left\{ \sigma \in \{0, 1\}^{|w|^k + c} \mid (v_0, p_1, \dots, v_{i-1}, p_i) \sqsubseteq P_{\langle P, V \rangle_\sigma}(w) \wedge \langle P, V \rangle_\sigma(w) = 1 \right\} \right| \\ W_i(w, v_0, p_1, \dots, v_{i-1}, p_i, v_i) &:= \\ &\max_P \left| \left\{ \sigma \in \{0, 1\}^{|w|^k + c} \mid (v_0, p_1, \dots, v_{i-1}, p_i, v_i) \sqsubseteq P_{\langle P, V \rangle_\sigma}(w) \wedge \langle P, V \rangle_\sigma(w) = 1 \right\} \right|. \end{aligned}$$

## 2 Interaktive Beweissysteme

Die Funktion  $Q_i(w, v_0, p_1, \dots, v_{i-1}, p_i)$  gibt die Anzahl von Zufallsfolgen  $\sigma$  an, für die ein mit den vorgegebenen Nachrichten der ersten  $i$  Runden beginnender interaktiver Beweis für das Wort  $w$  durch einen optimalen Beweiser  $P$  so fortgesetzt werden kann, dass der zum Entscheidungsproblem  $L$  gehörige Verifizierer  $V$  das Wort  $w$  schließlich akzeptiert.

Analog gibt die Funktion  $W_i(w, v_0, p_1, \dots, v_{i-1}, p_i, v_i)$  die entsprechende Anzahl von Zufallsfolgen  $\sigma$  an, wenn die Nachrichten der ersten  $i$  Runden zuzüglich der Nachricht des Verifizierers der  $(i+1)$ -ten Runde festgelegt ist.

Das folgende Lemma stellt einen Zusammenhang zwischen diesen Funktionen her, der zur rekursiven Berechnung der gesuchten Wahrscheinlichkeit ausgenutzt werden kann.

**Lemma 2.2.3.** *Sei  $L \in \text{IP}$ , jedoch o. B. d. A.  $\mathcal{P} = \mathcal{V} = \{0, 1\}$ . Dann gilt für einen geeigneten Verifizierer  $V$  mit einer o. B. d. A. nur von  $|w|$  abhängigen Rundenzahl  $r \leq |w|^k + c$*

1.  $w \in L \Leftrightarrow Q_0(w) \geq \left(\frac{1}{2} + \varepsilon\right) \cdot 2^{|w|^k + c}$ ,
2.  $Q_i(w, v_0, \dots, p_i) = \sum_{v_i \in \{0,1\}} W_i(w, v_0, \dots, p_i, v_i)$  (für  $i = 0, \dots, r$ ),
3.  $W_i(w, v_0, \dots, v_i) = \max_{p_{i+1} \in \{0,1\}} Q_i(w, v_0, \dots, v_i, p_{i+1})$  (für  $i = 0, \dots, r-1$ ) und
4.  $W_r(w, v_0, \dots, v_r) = \left| \left\{ \sigma \in \{0, 1\}^{|w|^k + c} \mid (v_0, p_1, \dots, v_r) = P_{\langle P, V \rangle_\sigma}(w) \wedge v_r = 1 \right\} \right|$ .

*Beweis.* Es folgen die Beweise der einzelnen Behauptungen.

1. Die Behauptung folgt unmittelbar aus den Definitionen.
2. Für einen festen Beweiser  $P'$  und feststehende erste  $i$  Runden gilt allgemein

$$\left| \left\{ \sigma \in \{0, 1\}^{|w|^k + c} \mid (v_0, \dots, v_{i-1}, p_i) \sqsubseteq P_{\langle P', V \rangle_\sigma}(w) \wedge \langle P', V \rangle_\sigma(w) = 1 \right\} \right| = \sum_{v_i \in \{0,1\}} \left| \left\{ \sigma \in \{0, 1\}^{|w|^k + c} \mid (v_0, \dots, v_{i-1}, p_i, v_i) \sqsubseteq P_{\langle P', V \rangle_\sigma}(w) \wedge \langle P', V \rangle_\sigma(w) = 1 \right\} \right|.$$

Es genügt also zu zeigen, dass ein Beweiser  $P$  existiert, der nach Ablauf der ersten  $i$  Runden abhängig von der Wahl der nächsten Nachricht  $v_i \in \{0, 1\}$  das jeweilige Maximum  $W_i(v_0, \dots, v_i)$  realisiert. Seien  $P_0$  und  $P_1$  die Beweiser, die für  $v_i = 0$  und  $v_i = 1$  diese Maxima realisieren. Da die Strategie eines Beweisers frei gewählt werden kann, sei  $P$  der Beweiser, der sich nach der  $i$ -ten Runde wie  $P_0$  verhält, falls  $v_i = 0$ , sonst wie  $P_1$ . Offensichtlich realisiert  $P$  für beide Fälle das jeweilige Maximum.

3. Die Menge der Beweiser für feststehende erste  $i$  Runden lässt sich partitionieren in die Mengen von Beweisern, die auf die folgende Nachricht  $v_i$  des Verifizierers mit  $p_{i+1} = 0$  bzw. mit  $p_{i+1} = 1$  antworten. Dementsprechend kann die Maximumsbildung separat für beide Partitionsklassen vorgenommen und anschließend von beiden das größere gewählt werden.

4. Da gemäß der Definition von  $W_i$  für  $i = r$  durch die Festlegung der  $p_1, \dots, p_r$  der Beweiser  $P$  eindeutig bestimmt ist, entfällt die Maximumsbildung und die Behauptung folgt.  $\square$

Der Wert der Funktion  $Q_0$  für das Wort  $w$  kann gemäß Lemma 2.2.3 rekursiv berechnet werden. Die Tiefe der Rekursion ist durch die Anzahl der Runden  $r$  des zugehörigen interaktiven Beweissystems polynomiell beschränkt. Am Ende jedes der exponentiell vielen Rekursionsabstiege muss der Wert der Funktion  $W_r$  berechnet werden. Dazu ist für ebenfalls exponentiell viele Zufallsfolgen  $\sigma$  die Auswertung des folgenden Prädikats nötig.

$$(v_0, p_1, \dots, v_r) = P_{\langle P, V \rangle_\sigma}(w) \wedge v_r = 1$$

Das Prädikat ist genau dann erfüllt, wenn der Verifizierer  $V$  für die jeweilige Zufallsfolge  $\sigma$  bei den angenommenen Antworten  $p_1, \dots, p_r$  eines Beweisers  $P$  den angegebenen interaktiven Beweis realisiert und das Wort  $w$  akzeptiert. Dies ist deterministisch und in polynomieller Zeit durch eine Simulation der zum Verifizierer  $V$  gehörigen Maschine  $M$  möglich.

Ein deterministisches Entscheidungsverfahren für ein Entscheidungsproblem  $L \in \text{IP}$  besteht demnach aus zwei Komponenten: einer vom Verifizierer unabhängigen zur Organisation der rekursiven Auswertung von  $Q_0$  und einer vom Verifizierer abhängigen zur Auswertung von  $W_r$ . Entsprechend ist  $L$  in denjenigen deterministischen Komplexitätsklassen enthalten, welche die Auswertung des zu  $W_r$  gehörigen Prädikats für exponentiell viele Rekursionsabstiege erlauben.

Diese Anforderungen werden durch die Komplexitätsklasse PSPACE erfüllt.

**Satz 2.2.4.** *Es gilt  $\text{IP} \subseteq \text{PSPACE}$ .*  $\square$

## 2.3 co-NP $\subseteq$ IP

Zu Beginn dieses Kapitels wurde beispielhaft gezeigt, dass das Komplement des Graphisomorphie-Problems  $\overline{\text{GI}}$  effiziente interaktive Beweise erlaubt. Somit ist gemäß der Vermutung  $\overline{\text{GI}} \notin \text{NP}$  anzunehmen, dass die Komplexitätsklasse IP auch über die Klasse NP hinausgehende Entscheidungsprobleme aus co-NP enthält.

Dieses Ergebnis wird in den folgenden Abschnitten zur Inklusion  $\text{co-NP} \subseteq \text{IP}$  erweitert. Dazu genügt es für ein co-NP-vollständiges Entscheidungsproblem einen Verifizierer anzugeben um somit die Existenz effizienter interaktiver Beweise für dieses Problem zu belegen. Die dafür nachfolgend eingeführte Beweistechnik wird in Abschnitt 2.4 für den Beweis  $\text{PSPACE} \subseteq \text{IP}$  wiederverwendet.

Ein für den folgenden Beweis geeignetes co-NP-vollständiges Problem ist das Komplement des klassischen Erfüllbarkeitsproblem aussagenlogischer Formeln in konjunktiver Normalform mit drei Literalen je Klausel  $\overline{3\text{SAT}}$ .

## 2 Interaktive Beweissysteme

**Definition 2.3.1.** Eine aussagenlogische Formel  $\phi$  über den Variablen  $\bar{x}_1, \dots, \bar{x}_n$  ist genau dann in *konjunktiver Normalform mit drei Literalen je Klausel*, in Zeichen 3KNF, wenn  $\phi$  die folgende Form hat. Dabei sei  $m$  die Anzahl der Klauseln von  $\phi$ .

- (i)  $\phi = \bigwedge_{i \in \{1, \dots, m\}} (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$
- (ii)  $\ell_{i,j} = \bar{x}_k$  oder  $\ell_{i,j} = \neg \bar{x}_k$  für ein  $k \in \{1, \dots, n\}$
- (iii)  $\ell_{i,1}, \ell_{i,2}, \ell_{i,3}$  sind paarweise verschieden

Die Grundidee für die Konstruktion eines effizienten interaktiven Beweissystems liegt dabei darin, die als unerfüllbar nachzuweisende aussagenlogische Formel  $\phi$  zunächst in eine arithmetische Formel  $\Phi$  zu überführen. Die Arithmetisierung wird so vorgenommen, dass  $\Phi$  genau für die nichterfüllenden Belegungen von  $\phi$  zu 0 ausgewertet wird. Entsprechend ergibt sich durch Summation der arithmetischen Formel über alle Belegungen genau dann 0 als Ergebnis, wenn die aussagenlogische Formel  $\phi$  unerfüllbar ist. Das Verschwinden dieser Summe wird schließlich durch einen interaktiven Beweis nachgewiesen.

### 2.3.1 Arithmetisierung von $\overline{3SAT}$ -Instanzen

Für aussagenlogische Formeln in einer Form gemäß Definition 2.3.1 sei die zugehörige arithmetisierte Formel wie folgt definiert.

**Definition 2.3.2.** Die *Arithmetisierung* einer aussagenlogischer Formel  $\phi$  über den Variablen  $\bar{x}_1, \dots, \bar{x}_n$  in 3KNF ist gegeben durch das nachfolgend definierte  $n$ -variate Polynom  $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}$  über den Variablen  $x_1, \dots, x_n$ .

- (i)  $\Phi(x_1, \dots, x_n) = \prod_i \sum_j \hat{\ell}_{i,j}$
- (ii)  $\hat{\ell}_{i,j} = x_k$ , falls  $\ell_{i,j} = \bar{x}_k$
- (iii)  $\hat{\ell}_{i,j} = (1 - x_k)$ , falls  $\ell_{i,j} = \neg \bar{x}_k$

Der Wahrheitswert FALSCH wird durch 0 und der Wahrheitswert WAHR durch 1 repräsentiert. Umgekehrt werden 0 als FALSCH und Werte  $> 0$  als WAHR interpretiert.

Als Arithmetisierung einer beliebigen aussagenlogischen Formel  $\phi$  in 3KNF ergibt sich ein Polynom dessen Grad höchstens der Anzahl der Klauseln in  $\phi$  entspricht.

**Beispiel 2.3.3.** Die folgende aussagenlogische Formel ist in 3KNF gegeben

$$\phi(\bar{x}_1, \bar{x}_2, \bar{x}_3) = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$

und hat die Arithmetisierung

$$\begin{aligned} \Phi(x_1, x_2, x_3) &= (x_1 + x_2 + x_3) \cdot (1 - x_1 + 1 - x_2 + 1 - x_3) \\ &= -x_1^2 - x_2^2 - x_3^2 - 2x_1x_2 - 2x_1x_3 - 2x_2x_3 + 3x_1 + 3x_2 + 3x_3. \end{aligned}$$

Eine nichterfüllende Belegung von  $\phi$  führt zu wenigstens einer unerfüllten Klausel, deren sämtliche Literale jeweils den Wahrheitswert FALSCH annehmen. Das bedeutet für die arithmetisierte Form, dass mit den einzelnen Literalen auch die Klausel als Summe der Literale zu 0 ausgewertet wird. Da die arithmetisierten Klauseln als Faktoren in die Formel  $\Phi$  eingehen, wird diese ebenfalls zu 0 ausgewertet.

Umgekehrt ist eine Belegung nur dann erfüllend, wenn in allen Klauseln mindestens ein Literal zu WAHR ausgewertet wird. Damit ist in der arithmetisierten Formel jede Klausel, sowie das Produkt aller Klauseln größer als 0. Somit folgt für äquivalente Belegungen der  $\bar{x}_i$  und  $x_i$

$$\phi(\bar{x}_1, \dots, \bar{x}_n) = \text{FALSCH} \Leftrightarrow \Phi(x_1, \dots, x_n) = 0.$$

Da eine aussagenlogische Formel genau dann unerfüllbar ist, wenn keine erfüllende Belegung der Variablen existiert, kann die Unerfüllbarkeit durch folgende Quantifizierung ausgedrückt werden:

$$\phi \notin \text{3SAT} \Leftrightarrow (\exists \bar{x}_1 \cdots \exists \bar{x}_n : \phi(\bar{x}_1, \dots, \bar{x}_n) = \text{WAHR}) = \text{FALSCH}.$$

Durch die Arithmetisierung des Existenzquantors als Summation über die möglichen Variablenbelegungen, analog einer disjunktiven Verknüpfung, ergibt sich

$$\phi \notin \text{3SAT} \Leftrightarrow \sum_{x_1 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(x_1, \dots, x_n) = 0.$$

Da  $\phi$  in konjunktiver Normalform mit drei Literalen je Klausel gegeben ist, nimmt die obige Summe maximal den numerischen Wert  $2^n \cdot 3^m$  an, wobei  $m$  die Anzahl der Klauseln in  $\phi$  ist. Daher kann die Summe modulo einer Primzahl  $p > 2^n \cdot 3^m$  ausgewertet werden und es folgt:

**Lemma 2.3.4.** Für eine aussagenlogische Formel  $\phi$  in 3KNF und für die zugehörige Arithmetisierung  $\Phi$  gilt mit einer wie oben gewählten Primzahl  $p$

$$\phi \notin \text{3SAT} \Leftrightarrow \sum_{x_1 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(x_1, \dots, x_n) = 0 \pmod{p}. \quad \square$$

Durch die Arithmetisierung reduziert sich die Frage nach der Unerfüllbarkeit einer aussagenlogischen Formel darauf, ob ein bestimmter arithmetischer Ausdruck 0 ergibt. Eine Fragestellung der Logik wird auf eine algebraische Fragestellung reduziert. Dies ermöglicht den Einsatz von Methoden aus einem anderen Bereich der Mathematik, um das ursprüngliche Problem zu lösen.

Aus dem Gaußschen Fundamentalsatz der Algebra folgt, dass ein univariates, nichttriviales Polynom mit Grad  $m$  über einem endlichen Zahlkörper höchstens  $m$  Nullstellen besitzt und daher zwei solche Polynome an höchstens  $m$  Stellen übereinstimmen können. Dieser Sachverhalt wird später für den Korrektheitsbeweis des interaktiven Beweissystems genutzt.

### 2.3.2 Interaktives Beweissystem für $\overline{3SAT}$

Die Aufgabe eines Verifizierers für  $\overline{3SAT}$  ist es, für eine aussagenlogische Formel  $\phi$  in 3KNF zu prüfen, ob  $\phi \notin 3SAT$  gilt. Gemäß Lemma 2.3.4 ist dies äquivalent zu

$$\sum_{x_1 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(x_1, \dots, x_n) = 0 \pmod{p}.$$

Die direkte Auswertung dieses Ausdrucks übersteigt die Möglichkeiten eines Verifizierers, da hierzu exponentiell viele Summanden zu berechnen und aufzuaddieren sind. Für den Nachweis des Verschwindens der Summe kann sich dieser stattdessen die Mächtigkeit der Beweiser zunutze zu machen.

Da Beweiser keiner polynomiellen Beschränkung der Laufzeit unterliegen, können sie den Wert der Summe direkt berechnen. Jedoch darf sich ein Verifizierer nicht vollkommen auf sie verlassen, da er sich sonst im Fall einer erfüllbaren Formel  $\phi$  leicht von einem betrügerischen Beweiser täuschen lassen würde.

Der im Folgenden angegebene Verifizierer  $V$  überführt betrügerische Beweiser mit hoher Wahrscheinlichkeit. Um die Zuverlässigkeit des Beweisers insgesamt auf die Probe stellen zu können, behält er sich einen Teil der Berechnungsarbeit selbst vor. Dazu verknüpft er die Behauptung des Beweisers über das Verschwinden der Summe mittels einer Sequenz von Kontrollrechnungen mit einer Entscheidungsrechnung, zu deren Ausführung der Verifizierer selbst im Stande ist.

---

#### Algorithmus 2.2 : IP-Verifizierer $V$ für $\overline{3SAT}$

---

**Eingabe** : aussagenlogische Formel  $\phi$  über Variablen  $\bar{x}_1, \dots, \bar{x}_n$  mit  $m$  Klauseln

**Ausgabe** : akzeptiere, falls  $\phi \in \overline{3SAT}$ , sonst verwerfe

**Daten** : Zufallszahlen  $r_1, \dots, r_n \in \{0, \dots, p-1\}$ ,

Zwischenergebnisse  $v_0, \dots, v_n \in \{0, \dots, p-1\}$

empfange Zahl  $p < 2^{n+1} \cdot 3^m$  vom Beweiser

prüfe, ob  $p > 2^n \cdot 3^m$  und  $p$  prim, sonst verwerfe

initialisiere  $v_0 := 0$

**for**  $i := 1$  **to**  $n$  **do**

empfange univariates Polynom  $\widehat{P}_i$  mit Grad  $\leq m$  vom Beweiser

prüfe, ob  $\widehat{P}_i(0) + \widehat{P}_i(1) = v_{i-1} \pmod{p}$ , sonst verwerfe

wähle  $r_i :=$  Zufallszahl aus  $\{0, \dots, p-1\}$

sende  $r_i$  an den Beweiser

berechne  $v_i := \widehat{P}_i(r_i) \pmod{p}$

berechne aus  $\phi$  das Polynom  $\Phi$

prüfe, ob  $\Phi(r_1, \dots, r_n) = v_n \pmod{p}$ , sonst verwerfe

akzeptiere

---

**Satz 2.3.5.** *Es gilt  $\overline{3SAT} \in \text{IP}$ .*

*Beweis.* Es genügt zu zeigen, dass der Verifizierer  $V$  aus Algorithmus 2.2 bei Eingabe einer Formel  $\phi \notin 3SAT$  mit hoher Wahrscheinlichkeit akzeptiert und umgekehrt bei  $\phi \in 3SAT$  mit hoher Wahrscheinlichkeit verwirft.

Dazu wird im Folgenden gezeigt, dass (i) im Fall  $\phi \notin 3SAT$  ein Beweiser existiert, der den Verifizierer  $V$  dazu bringt mit Wahrscheinlichkeit 1 zu akzeptieren. Weiter wird gezeigt, dass (ii) im Fall  $\phi \in 3SAT$  jeder Beweiser mit hoher Wahrscheinlichkeit daran scheitern muss, den Verifizierer  $V$  zum Akzeptieren zu bewegen. Abschließend wird gezeigt, dass (iii) der Verifizierer  $V$  mit polynomieller Laufzeit in der Länge der Formel  $\phi$  auskommt.

(i) sei  $\phi \notin 3SAT$

Das Akzeptierungsverhalten des Verifizierers  $V$  wird entscheidend durch die vom Beweiser gesendeten Polynome  $\widehat{P}_i$  bestimmt. Als solche eignen sich z. B. die wie folgt für alle  $i \in \{1, \dots, n\}$  und für die bis einschließlich zum  $(i-1)$ -ten Schleifendurchlauf an den Beweiser gesendeten Zufallszahlen  $r_1, \dots, r_{i-1}$  definierten univariaten Polynome:

$$P_i(x) := \sum_{x_{i+1} \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(r_1, \dots, r_{i-1}, x, x_{i+1}, \dots, x_n).$$

Damit der Verifizierer schließlich akzeptiert, muss insbesondere im ersten Durchlauf der Schleife für  $i = 1$  die Kontrollrechnung  $\widehat{P}_1(0) + \widehat{P}_1(1) = v_0 := 0$  bestanden werden. Da nach Voraussetzung  $\phi$  unerfüllbar ist, kann der Beweiser das Polynom  $\widehat{P}_1 := P_1$  an den Verifizierer senden. Dieses Polynom wird definitionsgemäß den Test bestehen:

$$\begin{aligned} \widehat{P}_1(0) + \widehat{P}_1(1) &= P_1(0) + P_1(1) \\ &= \sum_{x_1 \in \{0,1\}} \left( P_1(x_1) \right) \\ &= \sum_{x_1 \in \{0,1\}} \left( \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(x_1, \dots, x_n) \right) \\ &= 0 = v_0. \end{aligned}$$

Dies ist eine erste Bestätigung der Behauptung des Beweisers, dass  $\phi \notin 3SAT$ . Allerdings kann auch ein betrügender Beweiser ein Polynom  $\widehat{P}_1$  liefern, das diesen Test besteht. Deshalb wird in jedem weiteren Schleifendurchlauf  $i \in \{2, \dots, n\}$  das aktuelle Polynom  $\widehat{P}_i$  auf Verträglichkeit mit dem zuvor gesendeten Polynom  $\widehat{P}_{i-1}$  getestet. Dazu merkt sich der Verifizierer im  $(i-1)$ -ten Durchlauf den Wert  $v_{i-1} := \widehat{P}_{i-1}(r_{i-1})$  an der nach Empfang des Polynoms  $\widehat{P}_{i-1}$  zufällig bestimmten Stelle  $r_{i-1}$ .

## 2 Interaktive Beweissysteme

Da nach Voraussetzung  $\phi$  unerfüllbar ist, kann der Beweiser weiterhin das Polynom  $\widehat{P}_i := P_i$  an den Verifizierer senden, um die Kontrollrechnungen zu bestehen:

$$\begin{aligned}
 \widehat{P}_i(0) + \widehat{P}_i(1) &= P_i(0) + P_i(1) \\
 &= \sum_{x_i \in \{0,1\}} \left( \sum_{x_{i+1} \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(r_1, \dots, r_{i-1}, x_i, x_{i+1}, \dots, x_n) \right) \\
 &= \sum_{x_i \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(r_1, \dots, r_{i-2}, r_{i-1}, x_i, \dots, x_n) \\
 &= P_{i-1}(r_{i-1}) \\
 &= \widehat{P}_{i-1}(r_{i-1}) = v_{i-1}.
 \end{aligned}$$

Ebenso besteht der Beweiser in diesem Fall die abschließende Entscheidungsrechnung, da nach Definition  $v_n := \widehat{P}_n(r_n) = P_n(r_n) = \Phi(r_1, \dots, r_n)$  gilt. Daher wird der Verifizierer mit Wahrscheinlichkeit 1 akzeptieren, falls  $\phi \notin \text{3SAT}$ .

(ii) sei  $\phi \in \text{3SAT}$

Ein betrügender Beweiser muss zu Beginn ein von  $P_1$  verschiedenes Polynom  $\widehat{P}_1$  senden, da der Verifizierer sonst sofort wegen der Erfüllbarkeit von  $\phi$ , also wegen  $\widehat{P}_1(0) + \widehat{P}_1(1) = P_1(0) + P_1(1) \neq 0 = v_0$  verwerfen würde. Daher sieht er sich in den folgenden Schleifendurchläufen mit zwei widersprüchlichen Aufgaben konfrontiert:

- er muss ein Polynom  $\widehat{P}_i$  senden, das seine vorangegangene Lüge  $\widehat{P}_{i-1} \neq P_{i-1}$  stützt, für das also  $\widehat{P}_i(0) + \widehat{P}_i(1) = \widehat{P}_{i-1}(r_{i-1}) = v_{i-1}$  gilt,
- und er muss darauf hin arbeiten, die Entscheidungsrechnung zu bestehen, also spätestens im letzten Schleifendurchlauf für  $i = n$  ein Polynom  $\widehat{P}_i$  zu senden, für das  $v_i := \widehat{P}_i(r_i) = P_i(r_i)$  gilt.

Angenommen, er hat im vorherigen Durchlauf gelogen, also ein Polynom  $\widehat{P}_{i-1} \neq P_{i-1}$  gesendet, so dass mit der vom Verifizierer ebenfalls in der vorherigen Runde gewählten Zufallszahl  $r_{i-1}$  folgt, dass  $v_{i-1} := \widehat{P}_{i-1}(r_{i-1}) \neq P_{i-1}(r_{i-1})$ . Dann gilt, falls er das nicht gelogene Polynom  $\widehat{P}_i := P_i$  sendet:

$$\begin{aligned}
 \widehat{P}_i(0) + \widehat{P}_i(1) &= P_i(0) + P_i(1) \\
 &= P_{i-1}(r_{i-1}) \\
 &\neq \widehat{P}_{i-1}(r_{i-1}) = v_{i-1},
 \end{aligned}$$

und die Kontrollrechnung in Runde  $i$  scheitert. Er kann also nur dann das nicht gelogene Polynom  $\widehat{P}_i := P_i$  senden, wenn zufällig  $v_{i-1} := \widehat{P}_{i-1}(r_{i-1}) = P_{i-1}(r_{i-1})$  gilt. Da die beiden Polynome  $\widehat{P}_{i-1}$  und  $P_{i-1}$  vom Grad  $m$  nach Annahme ungleich sind, können sie an höchstens  $m$  Stellen übereinstimmen. Somit gilt wegen der Wahl von  $r_{i-1} \in \{0, \dots, p-1\}$ :

$$\Pr \left[ \widehat{P}_i(0) + \widehat{P}_i(1) = v_{i-1} \mid \widehat{P}_{i-1} \neq P_{i-1} \wedge \widehat{P}_i = P_i \right] = \frac{m}{p}.$$

Auch die abschließende Entscheidungsrechnung wird für ein  $\widehat{P}_n \neq P_n$  wegen  $\Phi(r_1, \dots, r_n) = P_n(r_n)$  nur mit folgender Wahrscheinlichkeit gelingen:

$$\Pr[\Phi(r_1, \dots, r_n) = v_n \mid \widehat{P}_n \neq P_n] = \frac{m}{p}.$$

Da der betrügende Beweiser im ersten Schleifendurchlauf ein gelogenes Polynom  $\widehat{P}_1 \neq P_1$  senden musste, bleiben ihm, um den Verifizierer  $V$  fälschlicherweise zum Akzeptieren zu bewegen, die folgende beiden Möglichkeiten:

- entweder der Beweiser empfängt in genau einer der Runden  $1 \leq i \leq n-1$  vom Verifizierer ein  $r_i$ , das ihm erlaubt in der folgenden Runde ein Polynom  $\widehat{P}_{i+1} := P_{i+1}$  zu senden, um später die Entscheidungsrechnung sicher zu bestehen,
- oder das in Runde  $n$  gesendete Polynom  $\widehat{P}_n \neq P_n$  erfüllt  $\widehat{P}_n(r_n) = P_n(r_n)$  und besteht anschließend die Entscheidungsrechnung.

Ein betrügender Beweiser  $P$  hat also in jeder der  $n$  Runden einen Versuch den Verifizierer  $V$  mit einer Wahrscheinlichkeit von  $\frac{m}{p}$  dazu zu bringen, schließlich zu akzeptieren. Die Täuschung gelingt daher mit einer Wahrscheinlichkeit  $\leq \frac{m}{p}$  in genau einer bestimmten der  $n$  Runden. Die Gesamtwahrscheinlichkeit, dass  $V$  fälschlicherweise akzeptiert, kann also mit Hilfe der Booleschen Ungleichung zu  $\leq \frac{nm}{p}$  abgeschätzt werden, und es gilt

$$\forall \text{ Beweiser } P: \Pr[\langle P, V \rangle(\phi) = 1 \mid \phi \notin \overline{3\text{SAT}}] \leq \frac{nm}{p}.$$

Bereits nach einmaliger Durchführung des interaktiven Beweises wird wegen der Wahl von  $p > 2^n \cdot 3^m$  eine exponentiell kleine Fehlerwahrscheinlichkeit bei erfüllbarer Formel  $\phi$  erzielt.

### (iii) effiziente Berechenbarkeit

Die vom Verifizierer  $V$  benötigte Laufzeit hängt entscheidend von der Größenordnung der Primzahl  $p$  ab. Da diese nach dem Bertrandschen Postulat aus dem Intervall  $[2^n \cdot 3^m, 2^{n+1} \cdot 3^m]$  gewählt werden kann, ist die Kodierungslänge von  $p$  polynomiell durch die Länge der Eingabe  $\phi$  begrenzt. Daher kann der Primzahltest für die vom Beweiser empfangene Zahl  $p$  mit probabilistischen [Rab76], beziehungsweise gemäß jüngeren Erkenntnissen [AKS04] auch mit deterministischen Verfahren effizient durchgeführt werden.

Weiter ist wegen der Ausführung der Berechnungen modulo der Primzahl  $p$  auch die Kodierungslänge der Zwischenergebnisse polynomiell beschränkt. Entsprechend können alle Berechnungen von einem polynomiell zeitbeschränkten Verifizierer durchgeführt werden.

Damit ist gezeigt, dass  $\overline{3\text{SAT}}$  effiziente interaktive Beweise erlaubt. Folglich gilt  $\overline{3\text{SAT}} \in \text{IP}$ .  $\square$

## 2 Interaktive Beweissysteme

Hieraus folgt wegen der co-NP-Vollständigkeit von  $\overline{3SAT}$  abschließend:

**Korollar 2.3.6.** *Es gilt  $co-NP \subseteq IP$ .* □

### 2.4 PSPACE $\subseteq$ IP

Zum Beweis der Inklusion  $PSPACE \subseteq IP$  genügt es, analog zum vorigen Beweis für  $co-NP \subseteq IP$ , einen Verifizierer anzugeben, der für ein PSPACE-vollständiges Problem effiziente interaktive Beweise realisieren kann. Als vollständiges Problem wird mit Q3SAT eine an die Mächtigkeit von PSPACE angepasste Variante des Erfüllbarkeitsproblems 3SAT genutzt.

Die zuvor am Beispiel co-NP eingeführte Beweistechnik liefert durch geeignete Anpassung der Arithmetisierung und Berücksichtigung daraus resultierender Detailprobleme unmittelbar einen interaktiven Beweis für Q3SAT. Zuvor folgt jedoch eine Beschreibung des Problems und ein Nachweis seiner Vollständigkeit.

#### 2.4.1 Q3SAT ist PSPACE-vollständig

Das Problem Q3SAT ist die Menge aller wahren, vollständig quantifizierten booleschen Formeln in pränexer und konjunktiver Normalform mit genau drei Literalen je Klausel. Die betrachteten Formeln haben also ohne Einschränkung die Form

$$\forall x_1 \exists x_2 \cdots Q_n: \phi(x_1, \dots, x_n),$$

wobei  $\phi$  in 3KNF gegeben ist und  $Q_n = \exists$ , falls  $n$  gerade, bzw.  $Q_n = \forall$ , falls  $n$  ungerade. Eine gegebene Formel  $\psi$  dieser Form ist genau dann wahr, wenn für alle durch die Quantifizierung vorgegebenen Belegungen der  $x_1, \dots, x_n$  die Teilformel  $\phi$  wahr ist. Die Definition von Q3SAT erinnert an die vollständigen Probleme der einzelnen Stufen der polynomiellen Hierarchie. Die Anzahl der Quantorenwechsel ist bei Q3SAT-Instanzen jedoch nicht auf eine Konstante festgelegt.

**Satz 2.4.1.** *Q3SAT ist PSPACE-vollständig.*

*Beweis.* Es ist relativ leicht zu sehen, dass  $Q3SAT \in PSPACE$  gilt. Hierzu genügt es zu überlegen, dass auf polynomiellem Platz mit einem rekursiven Verfahren alle Belegungen der quantifizierten Variablen ausprobiert werden können. Die Rekursion wird gesteuert durch das Ergebnis der Auswertung von  $\phi$  unter den einzelnen Belegungen.

Um den Nachweis der Vollständigkeit abzuschließen, ist für jede Sprache  $L \in PSPACE$  die Reduzierbarkeit von  $L$  auf Q3SAT nachzuweisen. Es genügt also, zu jedem Eingabewort  $w$  an eine beliebige aber feste PSPACE-Maschine  $M$  eine Q3SAT-Instanz anzugeben, die genau dann wahr ist, wenn  $w \in L(M)$  gilt.

Zu einer PSPACE-Maschine  $M$  gibt es ein Polynom  $q$ , so dass ihr Berechnungsergebnis für ein Eingabewort  $w$  nach höchstens  $2^{q(|w|)}$  Schritten feststeht und während der Berechnung maximal  $q(|w|)$  Platz verbraucht wird. Analog zur Beweisidee des Satzes von Cook [Coo71], wird eine einzelne Konfigurationen  $K$  von  $M$  als boolesche Formel polynomieller Länge kodiert. Da jedoch in einer Rechnung von  $M$  exponentiell viele Konfigurationen auftreten können, ist es nicht möglich die zugehörigen Übergänge in eine boolesche Formel polynomieller Länge zu kodieren.

Stattdessen wird eine quantifizierte boolesche Formel konstruiert, die genau dann wahr ist, wenn  $M$  mit der Eingabe  $w$  ausgehend von der Startkonfiguration  $K_{\text{Start}}(w)$  in höchstens  $2^{q(|w|)}$  Schritten zur o. B. d. A. eindeutigen Endkonfiguration  $K_{\text{Ende}}$  gelangt. Sei nun  $\psi_i(K_a, K_b)$  eine Formel, die genau dann wahr ist, wenn die Konfiguration  $K_b$  von  $K_a$  aus in höchstens  $2^i$  Schritten erreichbar ist. Dann gilt

$$w \in L(M) \Leftrightarrow \psi_{q(|w|)}(K_{\text{Start}}(w), K_{\text{Ende}}) = \text{WAHR}.$$

Die Konstruktion der  $\psi_i$  erfolgt rekursiv. Trivialerweise ist  $\psi_0(K_a, K_b)$  genau dann wahr, wenn  $K_a = K_b$  oder wenn  $K_b$  von  $K_a$  aus in einem Schritt erreichbar ist. Dies kann als boolesche Formel polynomieller Länge kodiert werden. Angemerkt sei, dass die Kodierungslänge von  $M$  selbst nur als Konstante in die Länge dieser Formel eingeht.

In einem Rekursionsschritt soll nun für  $i \geq 1$  eine quantifizierte boolesche Formel für  $\psi_i$  konstruiert werden. Eine naheliegende Möglichkeit ist es, nach dem Beweis zum Satz von Savitch [Sav70] vorzugehen und  $\psi_i(K_a, K_b)$  wie folgt zu konstruieren:

$$\psi_i(K_a, K_b) = \exists K_c: \left[ \psi_{i-1}(K_a, K_c) \wedge \psi_{i-1}(K_c, K_b) \right].$$

Die auf diese Weise konstruierte Formel für  $\psi_{q(|w|)}$  hätte allerdings exponentielle Länge, da sich in jedem der Rekursionsschritte von  $i = q(|w|)$  bis 1 die Anzahl der Formeln  $\psi_{i-1}$  gegenüber den  $\psi_i$  bis zum Rekursionsabbruch bei  $\psi_0$  verdoppelt. Ebenso fällt auf, dass die Formeln keine Allquantoren enthalten.

Das exponentielle Wachstum der Formellänge kann vermieden werden, wenn unter Verwendung von Allquantoren der Ausdruck  $\psi_{i-1}(K_a, K_c) \wedge \psi_{i-1}(K_c, K_b)$  wie folgt durch einen Ausdruck mit nur einem Rekursionsabstieg ersetzt wird:

$$\psi_i(K_a, K_b) = \exists K_c \forall K_x \forall K_y: \left[ ((K_x, K_y) = (K_a, K_c) \vee (K_x, K_y) = (K_c, K_b)) \Rightarrow \psi_{i-1}(K_x, K_y) \right].$$

Bei jedem Rekursionsschritt verlängert sich die Formel um eine additive Konstante, die allerdings selbst polynomielle Länge hat. Somit hat auch die für  $\psi_{q(|w|)}$  in  $q(|w|)$  Schritten konstruierte Formel polynomielle Länge.

Zuletzt bleiben noch zwei technische Maßnahmen, um die Formel in die korrekte Form zu überführen. Die pränex Normalform kann sehr leicht erreicht werden,

da alle Variablen vor ihrer Quantifizierung nicht und damit auch nicht ungebunden vorkommen. Somit können die Quantoren einfach an den Anfang der Formel verschoben werden. Zwischen zwei aufeinanderfolgenden Allquantoren wird ein Existenzquantor eingefügt, der eine extra eingeführte, nicht weiter benutzte Hilfsvariable quantifiziert.

Anschließend kann die hinter dem Quantorenblock verbliebene unquantifizierte boolesche Formel in 3KNF überführt werden. Dies gelingt, indem diese Formel in einen äquivalenten Schaltkreis übersetzt wird. Wie im Beweis für die NP-Vollständigkeit von circuit-SAT (Reduktion auf SAT, dann auf 3SAT, vgl. [Pap94]), wird der Schaltkreis in eine äquivalente boolesche Formel in konjunktiver Normalform mit genau drei Literalen je Klausel überführt.

Das beschriebene Konstruktionsverfahren liefert in polynomieller Zeit für eine beliebige aber feste PSPACE-Maschine  $M$  und eine Eingabe  $w$  eine Q3SAT-Instanz, die genau dann wahr ist, wenn die Eingabe  $w$  von der Maschine  $M$  akzeptiert wird. Somit ist Q3SAT PSPACE-vollständig.  $\square$

### 2.4.2 Arithmetisierung von Q3SAT-Instanzen

Bei der zuvor eingeführten Arithmetisierung von  $\overline{3SAT}$ -Instanzen war das Ziel, die Unerfüllbarkeit der booleschen Formel  $\phi$  möglichst einfach auf die arithmetische Formel  $\Phi$  zu übertragen. Insbesondere ergab sich  $\Phi(x_1, \dots, x_n) = 0$  für nichterfüllende Variablenbelegungen. Unberücksichtigt blieb, welchen Wert  $\Phi$  für erfüllende Belegungen annahm.

Bei Q3SAT ist die Erfüllbarkeit der Teilformel  $\phi$  unter den durch die Quantifizierung vorgegebenen Belegungen von Interesse. Die Arithmetisierung von  $\phi$  wird so gewählt, dass für erfüllende Belegungen  $\Phi(x_1, \dots, x_n) = 1$  gilt. Dies ermöglicht, die anschließende Arithmetisierung der Quantoren so vorzunehmen, dass insgesamt genau dann der Wert 1 angenommen wird, wenn die Q3SAT-Instanz wahr ist.

#### Arithmetisierung der unquantifizierten Teilformel

Für boolesche Teilformeln in einer Form gemäß Definition 2.3.1 sei die zugehörige arithmetisierte Formel wie folgt definiert.

**Definition 2.4.2.** Die *Arithmetisierung* einer booleschen Formel  $\phi$  über den Variablen  $\bar{x}_1, \dots, \bar{x}_n$  in 3KNF mit  $m$  Klauseln ist gegeben durch das nachfolgend definierte  $n$ -variate Polynom  $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}$  über den Variablen  $x_1, \dots, x_n$ .

$$(i) \quad \Phi(x_1, \dots, x_n) = \prod_{i \in \{1, \dots, m\}} (1 - (1 - \hat{\ell}_{i,1})(1 - \hat{\ell}_{i,2})(1 - \hat{\ell}_{i,3}))$$

$$(ii) \quad \hat{\ell}_{i,j} = x_k, \text{ falls } \ell_{i,j} = \bar{x}_k$$

(iii)  $\hat{\ell}_{i,j} = (1 - x_k)$ , falls  $\ell_{i,j} = \neg \bar{x}_k$

Der Wahrheitswert FALSCH wird durch 0 und der Wahrheitswert WAHR durch 1 repräsentiert. Umgekehrt werden auch 0 als FALSCH und 1 als WAHR interpretiert.

Als Arithmetisierung  $\Phi$  der Teilformel  $\phi$  ergibt sich ein  $n$ -variables Polynom mit Grad höchstens  $3m$ , wobei  $m$  die Anzahl der Klauseln von  $\phi$  angibt. Unter allen 0/1-Belegungen seiner Variablen nimmt  $\Phi$  genau für die Belegungen, die den erfüllenden Belegungen von  $\phi$  entsprechenden den Wert 1 an, sonst den Wert 0.

### Arithmetisierung der Quantoren

Eine einfache Variante die Quantoren zu arithmetisieren wäre es, die  $\forall x_i$ -Quantifizierungen durch  $\prod_{x_i \in \{0,1\}}$  und die  $\exists x_i$ -Quantifizierungen durch  $\sum_{x_i \in \{0,1\}}$  zu ersetzen. Die erhaltene Arithmetisierung könnte für wahre Formeln allerdings Werte in exponentieller Größenordnung annehmen. Genau wie zuvor bei der disjunktiven Verknüpfung der Literale einer Klausel muss auch hier die Arithmetisierung des Existenzquantors angepasst werden, um ausschließlich die Werte 0 oder 1 zu erhalten.

**Definition 2.4.3.** Für eine nicht vollständig quantifizierte boolesche Formel  $\psi$ , eine ihrer freien Variablen  $x$  und das aus  $\psi$  durch Arithmetisierung erhaltene Polynom  $P_\psi$  sei die *Arithmetisierung der Quantifizierung von  $x$  in  $\psi$*  durch folgende Polynome definiert

$$\begin{aligned} \forall x: \psi \quad \text{wird zu} \quad \prod_x P_\psi &:= P_\psi(x=0) \cdot P_\psi(x=1), \\ \exists x: \psi \quad \text{wird zu} \quad \prod_x P_\psi &:= P_\psi(x=0) + P_\psi(x=1) - P_\psi(x=0) \cdot P_\psi(x=1). \end{aligned}$$

Dabei sei  $P_\psi(x=\dots)$  die partielle Auswertung von  $P_\psi$  in der Variablen  $x$ . Mit einem Rekursionsabbruch für den Fall  $\psi = \phi$  und entsprechend  $P_\psi = \Phi$  ergibt sich somit für eine vollständig quantifizierte boolesche Formel  $\forall x_1 \exists x_2 \dots \exists x_n \phi$  (sei  $n$  o. B. d. A. gerade) die Arithmetisierung  $\prod_{x_1} \prod_{x_2} \dots \prod_{x_n} \Phi$ .

Die Anwendung der  $\prod_x$ - und  $\prod_x$ -Operationen auf ein Polynom verändert dessen Wertebereich bezüglich der 0/1-Belegungen seiner Variablen nicht, so dass für eine vollständig quantifizierte boolesche Formel und ihrer Arithmetisierung gilt

$$\forall x_1 \exists x_2 \dots \exists x_n \phi = \text{WAHR} \Leftrightarrow \prod_{x_1} \prod_{x_2} \dots \prod_{x_n} \Phi = 1.$$

Allerdings eignet sich diese Arithmetisierung noch nicht zur Verwendung für das interaktive Beweissystem. Durch die schrittweise Arithmetisierung ergeben sich

## 2 Interaktive Beweissysteme

Zwischenpolynome die über den noch nicht ausgewerteten Variablen exponentiellen Grad haben (Verdoppelung des Grades durch jede  $\prod_x$ - und  $\coprod_x$ -Operation). Die resultierenden Beschreibungen der im Allgemeinen dichten Polynome haben exponentielle Länge und können von einem polynomiell beschränkten Verifizierer weder empfangen noch verarbeitet werden.

Dieses Problem kann gelöst werden, indem nach jeder  $\prod_x$ - oder  $\coprod_x$ -Operation eine Folge von Operationen zur Reduktion des Grades der Zwischenpolynome eingeführt wird. Die Reduktion des Grades bezüglich einer Variable  $x$  gelingt, da unter den relevanten Belegungen  $x \in \{0, 1\}$  und für Exponenten  $k > 1$  stets  $x^k = x$  gilt. Daher sei die Operation  $\mathcal{R}_x$  zur Reduktion des Grades des Polynoms  $P$  bezüglich der Variable  $x$  wie folgt definiert. Dabei ist  $P(x = \dots)$  erneut die partielle Auswertung von  $P$  in  $x$ .

$$\begin{aligned}\mathcal{R}_x P &:= P(x=0) \cdot (1-x) + P(x=1) \cdot x \\ &= P \bmod (x^2 - x)\end{aligned}$$

Offensichtlich gilt  $(\mathcal{R}_x P)(x = \bar{x}) = P(x = \bar{x})$  für  $\bar{x} \in \{0, 1\}$ , d. h. die beiden Polynome  $\mathcal{R}_x P$  und  $P$  stimmen unter einer 0- bzw. 1-Belegung von  $x$  für beliebige Belegungen der übrigen Variablen überein. Der Grad von  $\mathcal{R}_x P$  ist bezüglich  $x$  auf 1 reduziert.

Durch Einfügen von Reduktionsoperationen für die noch nicht ausgewerteten Variablen ergibt sich eine Arithmetisierung mit geeignet beschränktem Grad der Zwischenpolynome und wiederum gilt genau für wahre Formeln:

$$\prod_{x_1} \mathcal{R}_{x_1} \prod_{x_2} \mathcal{R}_{x_1} \mathcal{R}_{x_2} \prod_{x_3} \dots \mathcal{R}_{x_1} \dots \mathcal{R}_{x_{n-1}} \prod_{x_n} \mathcal{R}_{x_1} \dots \mathcal{R}_{x_n} \Phi = 1.$$

Der Grad der Zwischenpolynome ist für die letzten  $n$  Reduktionsoperationen maximal  $3m$ , für die vorderen Reduktionsoperationen genau 2 und für alle übrigen Operationen genau 1. Da die Arithmetisierung ausschließlich die Werte 0 oder 1 annimmt, kann unter Berücksichtigung der maximalen Grade eine Primzahl  $p$  für die Auswertung der Polynome modulo  $p$  frei gewählt werden und es folgt:

**Lemma 2.4.4.** *Für eine quantifizierte boolesche Formel  $\forall x_1 \exists x_2 \dots \exists x_n \phi$  und die zugehörige abstrakt notierte Arithmetisierung gilt mit  $\ell = \sum_{i=1}^n (i+1)$ ,  $O_j \in \{\prod_{x_i}, \coprod_{x_i}, \mathcal{R}_{x_i} \mid i \leq n\}$  und einer geeignet gewählten Primzahl  $p$*

$$\forall x_1 \exists x_2 \dots \exists x_n \phi \in \text{Q3SAT} \Leftrightarrow O_1 O_2 \dots O_\ell \Phi = 1 \pmod{p}. \quad \square$$

### 2.4.3 Interaktives Beweissystem für Q3SAT

Wie zuvor beim Beweis für  $\text{co-NP} \subseteq \text{IP}$  besteht die Schwierigkeit für den Verifizierer darin, einerseits die PSPACE-Mächtigkeit des Beweisers auszunutzen und sich andererseits der Vertrauenswürdigkeit des Beweisers in hohem Maße sicher zu sein. Letzteres jedoch, ohne seine Berechnungen selbst nachvollziehen zu können.

Der Verifizierer für Q3SAT (Algorithmus 2.3) verwendet zum Erreichen beider Ziele eine Variante der zuvor vom  $\overline{3SAT}$ -Verifizierer realisierten Strategie. Die vorgenommene Anpassung besteht ausschließlich darin, die mit der Arithmetisierung eingeführten Operationen  $\prod_{x'}$ ,  $\prod_x$  und  $\mathcal{R}_x$  besonders zu berücksichtigen.

**Satz 2.4.5.** *Es gilt Q3SAT  $\in$  IP.*

*Beweis.* Nach Lemma 2.4.4 ist die Aufgabe des Verifizierers die Führung eines interaktiven Beweises für

$$O_1 O_2 \cdots O_\ell \Phi = 1.$$

Der Verifizierer kann die Formel nicht vollständig selbst auswerten, daher bittet er den Beweiser um Hilfe. Allerdings darf er sich nicht auf diesen verlassen, sondern übernimmt einen Teil der Auswertung selbst. So kann er die Vertrauenswürdigkeit des Beweisers auf die Probe stellen. Um sich also vom Beweiser für  $\Phi_0 := \Phi$  von

$$v_0 := 1 = O_1 O_2 \cdots O_\ell \Phi_0$$

überzeugen zu lassen, sendet dieser ein univariates Polynom  $\widehat{P}$  mit Grad entsprechend dem des erwarteten Zwischenpolynoms. Der Verifizierer nutzt  $\widehat{P}$  zu zwei Zwecken:

1. er prüft, ob  $v_0 = O_1 \widehat{P}$  und verwirft ansonsten,
2. er verlangt vom Beweiser, ihn für eine Zufallszahl  $r_1$  und  $\Phi_1 := \Phi(x_1 = r_1)$  von folgender Äquivalenz zu überzeugen:

$$v_1 := \widehat{P}(r_1) = O_2 \cdots O_\ell \Phi_1.$$

Es werden also in einem rekursiven Verfahren der Reihe nach die einzelnen Quantoren bzw. Operationen der Formel eliminiert.

Um sich auf diese Weise in Runde  $k$  von  $v_{k-1} = O_k \cdots O_\ell \Phi_{i-1}$  überzeugen zu lassen (mit  $O_k = \prod_{x_i'}$ ,  $\prod_{x_i}$  oder  $\mathcal{R}_{x_i}$  für ein  $i$  und  $\Phi_j := \Phi(x_1 = r_1, \dots, x_j = r_j)$ ), prüft der Verifizierer für das empfangene Polynom  $\widehat{P}$ , ob  $v_{k-1} = O_k \widehat{P}$ , wählt eine Zufallszahl  $r_i$  und ein passendes  $v_k := \widehat{P}(r_i)$  und beginnt dann die nächste Runde (eine genaue Beschreibung der Fallunterscheidung für die  $O_k$  folgt weiter unten).

Der Verifizierer kann schließlich im Anschluß an die letzte Runde die Äquivalenz

$$v_\ell := \widehat{P}(r_n) = \Phi_n$$

selbst überprüfen und gegebenenfalls akzeptieren.

Die Vollständigkeit und Korrektheit des Verfahrens ergibt sich ähnlich wie zuvor im Fall  $\text{co-NP} \subseteq \text{IP}$ . Anstelle der ausschließlichen Betrachtung der  $\sum_{x_i \in \{0,1\}}$ -Operation ist nun für die einzelnen Runden eine Fallunterscheidung für die möglichen Operationen  $\prod_{x_i'}$ ,  $\prod_{x_i}$  und  $\mathcal{R}_{x_i}$  nötig. So tritt in Runde  $k$  für  $1 \leq k \leq \ell$  und jeweils ein  $i$  mit  $1 \leq i \leq n$  einer der folgenden drei Fälle auf:

---

**Algorithmus 2.3** : IP-Verifizierer  $V$  für Q3SAT

---

**Eingabe** : quantifizierte boolesche Formel  $\psi = Q_1x_1Q_2x_2 \cdots Q_nx_n: \phi(x_1, \dots, x_n)$ ,  
 $\phi$  in 3KNF mit  $m$  Klauseln und  $Q_i = \forall$ , falls  $i$  ungerade, sonst  $Q_i = \exists$ ,  
 seien  $O_1, \dots, O_\ell$  die Operationen der Arithmetisierung von  $\psi$

**Ausgabe** : akzeptiere, falls  $\psi \in \text{Q3SAT}$ , sonst verwerfe

**Daten** : Zufallszahlen  $r_1, \dots, r_n \in \{0, \dots, p-1\}$ ,  
 Zwischenergebnis  $v \in \{0, \dots, p-1\}$

wähle Primzahl  $p > c \cdot (n^2 + 3mn)$  für ein  $c > 2$

initialisiere  $v := 1$

**for**  $k := 1$  **to**  $\ell = \sum_{i=1}^n (i+1)$  **do**

**switch**  $O_k$  **do**

**case**  $\prod_{x_i}$  (für ein  $i$ )

            empfange univariates Polynom  $\widehat{P}$  mit Grad 1 vom Beweiser  
             prüfe, ob  $\prod_{x_i} \widehat{P} = v \pmod{p}$ , sonst verwerfe

**case**  $\coprod_{x_i}$  (für ein  $i$ )

            empfange univariates Polynom  $\widehat{P}$  mit Grad 1 vom Beweiser  
             prüfe, ob  $\coprod_{x_i} \widehat{P} = v \pmod{p}$ , sonst verwerfe

**case**  $\mathcal{R}_{x_i}$  (für ein  $i$ )

            empfange univariates Polynom  $\widehat{P}$  mit Grad  $\begin{cases} \leq 3m & \text{falls } k > \ell - n \\ 2 & \text{falls } k < \ell - n \end{cases}$  vom  
             Beweiser  
             prüfe, ob  $(\mathcal{R}_{x_i} \widehat{P})(r_i) = v \pmod{p}$ , sonst verwerfe

    wähle  $r_i :=$  Zufallszahl aus  $\{0, \dots, p-1\}$

    sende  $r_i$  an den Beweiser

    berechne  $v := \widehat{P}(r_i) \pmod{p}$

berechne aus  $\phi$  das Polynom  $\Phi$

prüfe, ob  $\Phi(r_1, \dots, r_n) = v \pmod{p}$ , sonst verwerfe

akzeptiere

---

$O_k = \prod_{x_i}$ : Hier versucht der Beweiser den Verifizierer von der Äquivalenz

$$v_{k-1} = \prod_{x_i} \mathcal{R}_{x_1} \cdots \mathcal{R}_{x_i} \prod_{x_{i+1}} \cdots \prod_{x_n} \mathcal{R}_{x_1} \cdots \mathcal{R}_{x_n} \Phi_{i-1} \quad (2.1)$$

zu überzeugen. Dazu sendet er ein univariates Polynom  $\widehat{P}$  mit Grad 1, das der Verifizierer dem Test  $v_{k-1} = \prod_{x_i} \widehat{P}$  unterzieht. Um diese Runde abzuschließen, wählt der Verifizierer erstmals die Zufallszahl  $r_i \in \{0, \dots, p-1\}$ . Der Beweiser wird dann in der nächsten Runde versuchen, den Verifizierer von

$$v_k := \widehat{P}(r_i) = \mathcal{R}_{x_1} \cdots \mathcal{R}_{x_i} \prod_{x_{i+1}} \cdots \prod_{x_n} \mathcal{R}_{x_1} \cdots \mathcal{R}_{x_n} \Phi_i \quad (2.2)$$

zu überzeugen.

Angenommen Äquivalenz 2.1 ist erfüllt, so kann der Beweiser das Polynom

$$\widehat{P}(x_i) := P(x_i) := \mathcal{R}_{x_1} \cdots \mathcal{R}_{x_i} \prod_{x_{i+1}} \cdots \prod_{x_n} \mathcal{R}_{x_1} \cdots \mathcal{R}_{x_n} \Phi_{i-1}$$

senden und der Verifizierer wird in dieser Runde nicht verwerfen. Außerdem wird auch die Äquivalenz 2.2 für jede Wahl von  $r_i$  erfüllt sein.

Falls Äquivalenz 2.1 nicht erfüllt ist, muss der Beweiser ein  $\widehat{P} \neq P$  senden, damit der Verifizierer nicht sofort verwirft. Mit dieser Wahl ist aber Äquivalenz 2.2 nur mit Wahrscheinlichkeit  $\frac{1}{p}$  erfüllt, da  $\Pr[\widehat{P}(r_i) = P(r_i) \mid \widehat{P} \neq P] = \frac{1}{p}$ .

$O_k = \prod_{x_i}$ : Dieser Fall ähnelt dem vorigen und wird nicht gesondert betrachtet.

$O_k = \mathcal{R}_{x_i}$ : Hier versucht der Beweiser den Verifizierer von der Äquivalenz

$$v_{k-1} = \mathcal{R}_{x_i} \cdots \mathcal{R}_{x_{j-1}} O_{x_j} \cdots \prod_{x_n} \mathcal{R}_{x_1} \cdots \mathcal{R}_{x_n} \Phi_{j-1} \quad (2.3)$$

zu überzeugen (für ein  $j$  mit  $i < j \leq n$  und  $O = \prod$  bzw.  $\prod$ , analog für die  $n$  letzten  $\mathcal{R}$ -Operationen). Dazu sendet er ein univariates Polynom  $\widehat{P}$  mit Grad 2 bzw.  $3m$ , das vom Verifizierer mit der zuvor gewählten Zufallszahl  $r_i$  dem Test  $v_{k-1} = (\mathcal{R}_{x_i} \widehat{P})(r_i)$  unterzogen wird. Um diese Runde abzuschließen, wählt der Verifizierer eine neue Zufallszahl  $r_i \in \{0, \dots, p-1\}$ . Der Beweiser wird dann in der nächsten Runde versuchen, den Verifizierer von

$$v_k := \widehat{P}(r_i) = \mathcal{R}_{x_{i+1}} \cdots \mathcal{R}_{x_{j-1}} O_{x_j} \cdots \prod_{x_n} \mathcal{R}_{x_1} \cdots \mathcal{R}_{x_n} \Phi_{j-1} \quad (2.4)$$

zu überzeugen.

Angenommen Äquivalenz 2.3 ist erfüllt, so kann der Beweiser das wegen  $\Phi_j^i := \Phi(x_1=r_1, \dots, x_{i-1}=r_{i-1}, x_{i+1}=r_{i+1}, \dots, x_j=r_j)$  univariate Polynom

$$\widehat{P}(x_i) := P(x_i) := \mathcal{R}_{x_{i+1}} \cdots \mathcal{R}_{x_{j-1}} O_{x_j} \cdots \prod_{x_n} \mathcal{R}_{x_1} \cdots \mathcal{R}_{x_n} \Phi_{j-1}^i$$

## 2 Interaktive Beweissysteme

senden und der Verifizierer wird in dieser Runde nicht verwerfen. Außerdem wird auch Äquivalenz 2.4 für jede Wahl von  $r_i$  erfüllt sein.

Analog zum ersten Fall muss der Beweiser, falls Äquivalenz 2.3 nicht erfüllt ist, ein Polynom  $\widehat{P} \neq P$  senden, damit der Verifizierer nicht sofort verwirft. Mit dieser Wahl gilt aber, dass Äquivalenz 2.4 nur mit Wahrscheinlichkeit  $\frac{2}{p}$  bzw.  $\frac{3m}{p}$  erfüllt ist.

Die aus den einzelnen Runden resultierenden Wahrscheinlichkeiten lassen sich für die  $\prod$ - und  $\coprod$ -Operationen, die  $n$  letzten, sowie die übrigen  $\mathcal{R}$ -Operationen mittels der Booleschen Ungleichung zusammenfassen. Insgesamt resultiert als Abschätzung der Wahrscheinlichkeit für das fälschliche Akzeptieren einer unwahren quantifizierten booleschen Formel:

$$n \frac{1}{p} + n \frac{3m}{p} + \left( \sum_{i=1}^{n-1} i \right) \frac{2}{p} = \frac{n^2 + 3mn}{p}.$$

Es genügt die Wahl einer Primzahl  $p > c \cdot (n^2 + 3mn)$  für ein  $c > 2$  um die Fehlerwahrscheinlichkeit des Verifizierers bei unwahren Formeln auf  $\frac{1}{2} - \varepsilon$  mit  $\varepsilon > 0$  zu beschränken. Die Fehlerwahrscheinlichkeit bei wahren Formeln ist 0. Es gilt also:

$$\begin{aligned} \exists \text{ Beweiser } P^*: \Pr[\langle P^*, V \rangle(\psi) = 1 \mid \psi \in \text{Q3SAT}] &= 1 \quad \text{und} \\ \forall \text{ Beweiser } P: \Pr[\langle P, V \rangle(\psi) = 1 \mid \psi \notin \text{Q3SAT}] &\leq \frac{n^2 + 3mn}{p}. \end{aligned}$$

Darüber hinaus ist leicht zu überprüfen, dass der Verifizierer  $V$  tatsächlich in polynomieller Zeit akzeptiert oder verwirft. Damit ist  $\text{Q3SAT} \in \text{IP}$  gezeigt.  $\square$

Wegen der PSPACE-Vollständigkeit von Q3SAT und wegen  $\text{IP} \subseteq \text{PSPACE}$  folgt:

**Korollar 2.4.6.** *Es gilt  $\text{PSPACE} = \text{IP}$ .*  $\square$

*Bemerkung 2.4.7.* Es sei angemerkt, dass der Verifizierer  $V$  zusätzlich zu den Zufallszahlen und den empfangenen Nachrichten durch die Wahl von  $p$  nur einen logarithmischen Platzbedarf hat. Außerdem bestehen die von  $V$  gesendeten Nachrichten ausschließlich aus den gewählten Zufallszahlen.

In [Sha92] wird ein Verifizierer *schwach* (orig.: *weak*) genannt, wenn er außer den öffentlich gewählten Zufallszahlen keine weiteren Nachrichten sendet, logarithmischen Platzbedarf hat und perfekte Vollständigkeit aufweist. Die Zufallszahlen werden dabei auf einem zusätzlichen Nur-Lese-Speicher bereitgestellt und Nachrichten über einen separaten Kommunikationsspeicher ausgetauscht.

Da auch die zuvor angegebene Reduktion einer beliebigen PSPACE-Sprache auf Q3SAT auf logarithmischem Platz berechnet werden kann, genügen bereits schwache Verifizierer, um PSPACE vollständig zu charakterisieren.

## 2.5 Bibliographische Hinweise

Die Klasse  $IP$  wurde erstmals in [GMR89] definiert. Eine andere Definition interaktiver Beweissysteme (*Arthur-Merlin-Spiele*) wurde zeitgleich in [Bab85] gegeben. Bei dieser Definition sind die konkreten Zufallswerte öffentlich. Dagegen kann der Verifizierer eines  $IP$ -Systems diese geheim halten. Überraschenderweise wurde in [GS86] gezeigt, dass beide Definitionen äquivalent sind.

In [Bab90] wird ein humoristisch inspirierter Überblick über die Entwicklung von der Definition bis zum Beweis  $IP = PSPACE$  und darüber hinaus gegeben. Ein entscheidender Schritt war der in [LFKN92] geführte Beweis für  $PH \subseteq IP$ . Dieser wurde kurze Zeit später in [Sha92] zu  $PSPACE \subseteq IP$  vervollständigt. Die hier vorgestellte Variante dieses Beweises basiert auf der in [She92] angegebenen, vereinfachten Fassung.

Heute finden sich in zahlreichen Lehrbüchern und Vorlesungsskripten Kapitel zur Komplexitätsklasse  $IP$ . So gibt [BC94] einen vollständigen aber knappen Überblick über das Thema. In [AB08] und [Kat07] wird der Beweis ausführlich und verständlich aufbereitet wiedergegeben. Bei [DK00] liegt der Fokus auf interaktiven Beweisen in der Arthur-Merlin-Variante, es werden aber die gleichen Ergebnisse gezeigt. Zusätzlich wird in [Sip06] ein dem hier wiedergegebenen ähnlicher formaler Beweis für  $IP \subseteq PSPACE$  geführt. Eher fragmentarisch, dafür mit zahlreichen Übungsaufgaben, wird der Beweis für  $PSPACE \subseteq IP$  in [SP98] geführt.

Ein weitreichender Überblick der durch die Resultate zu interaktiven Beweissystemen angestoßenen Entwicklungen und der daraufhin erzielten Ergebnisse ist in [Bab94] zu finden.



## 3 Reelle interaktive Beweissysteme

Bei der Übertragung klassischer interaktiver Beweissysteme in das BSS-Berechnungsmodell entstehen nach Ivanov und de Rougemont [IdR98] unterschiedliche reelle Komplexitätsklassen. Diese sind charakterisiert durch den Typ der Nachrichten – reell oder diskret – die Verifizierer und Beweiser austauschen dürfen.

Offensichtlich macht es keinen Unterschied, ob der Verifizierer diskrete oder reelle Nachrichten sendet, da jede Nachricht eines Verifizierers deterministisch von der Eingabe, den Maschinenkonstanten und der konkreten Zufallsfolge abhängt. Somit kann der Beweiser die reellen Nachrichten des Verifizierers selbst berechnen, wenn dieser die diskrete Zufallsfolge als Nachricht überträgt.

Wird dem Beweiser gestattet reelle Nachrichten zu senden, so kann der Verifizierer die Mächtigkeit des Beweisers ausnutzen, um mit reellen Zahlen einer Größenordnung zu rechnen, die seine eigenen Möglichkeiten übersteigen. Die zugehörige Komplexitätsklasse reeller interaktiver Beweissysteme mit reellen Nachrichten wird durch  $IP_{\mathbb{R}}$  bezeichnet.

Bleibt der Beweiser auf diskrete Nachrichten beschränkt, so ergibt sich die Komplexitätsklasse  $BIP_{\mathbb{R}}$  der reellen interaktiven Beweissysteme mit diskreten Nachrichten. Durch diese Einschränkung entsteht eine echt kleinere Klasse, die allerdings im auf die Addition beschränkten BSS-Modell äquivalent zu einer reellen Entsprechung der klassischen Komplexitätsklasse PSPACE ist. Diese beiden Resultate sind die zentralen Ergebnisse in [IdR98] und werden in den Abschnitten 3.3 sowie 3.4 ausführlich vorgestellt.

Die zuvor folgenden Abschnitte 3.1 und 3.2 beziehen sich eng auf die entsprechenden Abschnitte in Kapitel 2. Es werden zunächst die reellen Komplexitätsklassen  $IP_{\mathbb{R}}$  und  $BIP_{\mathbb{R}}$  sowie ihre additiven Einschränkungen definiert. Anschließend folgt analog zum klassischen Fall der Nachweis der Inklusion  $BIP_{\mathbb{R}} \subseteq PAR_{\mathbb{R}}$ .

### 3.1 Formale Definition

Die in Abschnitt 2.1 eingeführten und vom konkreten Berechnungsmodell abstrahierenden Definitionen interaktiver Beweissysteme werden im Folgenden verwendet um die Komplexitätsklassen  $IP_{\mathbb{R}}$  und  $BIP_{\mathbb{R}}$  zu beschreiben. Anschließend werden die für die nachfolgenden Abschnitte relevanten additiven Varianten dieser und anderer Komplexitätsklassen aus Abschnitt 1.4 definiert.

### 3 Reelle interaktive Beweissysteme

**Definition 3.1.1.** Die Klasse  $\text{IP}_{\mathbb{R}}$  (*interactive polynomial-time*) ist für  $\mathcal{P} = \mathbb{R}$  sowie  $\mathcal{V} = \{0,1\}^*$  definiert als die Menge genau der Entscheidungsprobleme  $(Y, L)$  mit  $L \subseteq Y \subseteq \mathbb{R}^\infty$ , für die ein Verifizierer  $V$  über  $Y$  und eine Konstante  $\varepsilon > 0$  existieren, so dass folgende Bedingungen erfüllt sind.

- (i)  $\forall \sigma \in \{0,1\}^*: \langle P, V \rangle_\sigma$  ist effizient BSS-berechenbar über  $Y$
- (ii)  $\forall y \in L \quad \exists \text{Beweiser } P: \Pr_\sigma \left[ \langle P, V \rangle_\sigma(y) = 1 \right] \geq \frac{1}{2} + \varepsilon$
- (iii)  $\forall y \in Y \setminus L \quad \forall \text{Beweiser } P: \Pr_\sigma \left[ \langle P, V \rangle_\sigma(y) = 0 \right] \geq \frac{1}{2} + \varepsilon$

Die Eingaben  $y \in Y$ , für die gemäß (ii)  $y \in L$  gilt, werden als *vom Verifizierer  $V$  akzeptiert* bezeichnet. Die übrigen Eingaben  $y \in Y$ , für die gemäß (iii)  $y \in Y \setminus L$  gilt, werden dagegen als *vom Verifizierer  $V$  verworfen* bezeichnet. Der Verifizierer  $V$  entscheidet  $L$  über  $Y$ .

**Definition 3.1.2.** Die Klasse  $\text{BIP}_{\mathbb{R}}$  (*boolean interactive polynomial-time*) ist definiert als die Einschränkung der Klasse  $\text{IP}_{\mathbb{R}}$  auf  $\mathcal{P} = \mathcal{V} = \{0,1\}^*$ . Dementsprechend kann die reelle Komplexitätsklasse  $\text{BIP}_{\mathbb{R}}$  als Adaption der klassischen Komplexitätsklasse  $\text{IP}$  auf das BSS-Maschinenmodell betrachtet werden.

**Definition 3.1.3.** Die *additiven reellen Komplexitätsklassen*  $\text{IP}_{\mathbb{R},+}$ ,  $\text{BIP}_{\mathbb{R},+}$ ,  $\text{PAR}_{\mathbb{R},+}$  und  $\text{PSPACE}_{\mathbb{R},+}$  sind eingeschränkte Varianten der entsprechenden Klassen  $\text{IP}_{\mathbb{R}}$ ,  $\text{BIP}_{\mathbb{R}}$ ,  $\text{PAR}_{\mathbb{R}}$  und  $\text{PSPACE}_{\mathbb{R}}$ . Sie ergeben sich aus einer Beschränkung der zugrunde liegenden Berechnungsmodelle auf die algebraischen Operationen  $+$  und  $-$ . Im Unterschied dazu erlauben die uneingeschränkten Klassen auch die Multiplikationsoperation.

## 3.2 $\text{BIP}_{\mathbb{R}} \subseteq \text{PAR}_{\mathbb{R}}$

Der in Abschnitt 2.2 geführte Beweis für  $\text{IP} \subseteq \text{PSPACE}$  ist bis auf die abschließende Prüfung des Prädikats  $(v_0, p_1, \dots, v_r) = P_{\langle P, V \rangle_\sigma}(y) \wedge v_r = 1$  nicht abhängig vom verwendeten Berechnungsmodell. Da auch die interaktiven Beweise der Klasse  $\text{BIP}_{\mathbb{R}}$  nur diskrete Nachrichten austauschen gilt somit analog zu Lemma 2.2.3:

**Lemma 3.2.1.** Sei  $(Y, L) \in \text{BIP}_{\mathbb{R}}$ , jedoch o. B. d. A.  $\mathcal{P} = \mathcal{V} = \{0,1\}$ . Dann gilt für einen geeigneten Verifizierer  $V$  mit einer o. B. d. A. nur von  $|y|$  abhängigen Rundenzahl  $r \leq |y|^k + c$

1.  $y \in L \Leftrightarrow Q_0(y) \geq \left(\frac{1}{2} + \varepsilon\right) \cdot 2^{|y|^k + c}$ ,
2.  $Q_i(y, v_0, \dots, p_i) = \sum_{v_i \in \{0,1\}} W_i(y, v_0, \dots, p_i, v_i) \quad (\text{für } i = 0, \dots, r)$ ,
3.  $W_i(y, v_0, \dots, v_i) = \max_{p_{i+1} \in \{0,1\}} Q_i(y, v_0, \dots, v_i, p_{i+1}) \quad (\text{für } i = 0, \dots, r-1) \text{ und}$
4.  $W_r(y, v_0, \dots, v_r) = \left| \left\{ \sigma \in \{0,1\}^{|y|^k + c} \mid (v_0, p_1, \dots, v_r) = P_{\langle P, V \rangle_\sigma}(y) \wedge v_r = 1 \right\} \right|. \quad \square$

Die Bestimmung von  $Q_0(y)$  erfordert wegen der polynomiellen Beschränkung der Rundenzahl  $r$  die Berücksichtigung einer exponentiellen Anzahl von Rekursionsabstiegen und dabei jeweils die Auswertung des genannten vom Berechnungsmodell abhängigen Prädikats für alle  $\sigma \in \{0, 1\}^{|y|^{k+c}}$ . Beides gelingt im allgemeinen und im additiven BSS-Modell in paralleler polynomieller Zeit. Entsprechend der Bemerkung 1.4.13 folgt daher:

**Satz 3.2.2.** *Es gilt  $\text{BIP}_{\mathbb{R}} \subseteq \text{PAR}_{\mathbb{R}}$  sowie  $\text{BIP}_{\mathbb{R}_+} \subseteq \text{PAR}_{\mathbb{R}_+}$ .* □

### 3.3 $\text{PAR}_{\mathbb{R}_+} \subseteq \text{BIP}_{\mathbb{R}_+}$

Eines der zentralen Resultate von Ivanov und de Rougemont [IdR98] ist die Inklusion  $\text{PAR}_{\mathbb{R}_+} \subseteq \text{BIP}_{\mathbb{R}_+}$ . In den folgenden Abschnitten wird der entsprechende Beweis ausführlich nachvollzogen. Zusammen mit der zuvor bewiesenen Inklusion  $\text{BIP}_{\mathbb{R}_+} \subseteq \text{PAR}_{\mathbb{R}_+}$  ergibt sich als ein erstes, dem Resultat von Shamir ähnliches Ergebnis die Äquivalenz  $\text{BIP}_{\mathbb{R}_+} = \text{PAR}_{\mathbb{R}_+}$ .

Zum Beweis der Inklusion  $\text{PAR}_{\mathbb{R}_+} \subseteq \text{BIP}_{\mathbb{R}_+}$  genügt es, einen  $\text{BIP}_{\mathbb{R}_+}$ -Verifizierer für ein  $\text{PAR}_{\mathbb{R}_+}$ -vollständiges Problem anzugeben. Daher wird im Folgenden zunächst mit HDTRAO ein solches Problem zusammen mit dem entsprechenden Vollständigkeitsbeweis angegeben.

Der von Ivanov und de Rougemont angegebene  $\text{BIP}_{\mathbb{R}_+}$ -Verifizierer für HDTRAO löst das Entscheidungsproblem, indem er die folgenden beiden Teilaufgaben bewältigt:

1. Reduktion von HDTRAO auf das diskrete Entscheidungsproblem Q3SAT,
2. Entscheidung der erhaltenen Q3SAT-Instanz.

Die entscheidene Leistung liegt dabei im ersten Schritt, also in der Reduktion eines Entscheidungsproblems einer additiven reellen Komplexitätsklasse auf ein Entscheidungsproblem einer diskreten Komplexitätsklasse, das gemäß Shamirs Resultat bereits in IP und somit auch in  $\text{BIP}_{\mathbb{R}_+}$  enthalten ist.

Im Anschluss an den Vollständigkeitsbeweis für HDTRAO liegt der Schwerpunkt der folgenden Abschnitte daher auf einem Beweis der Existenz einer derartigen Reduktion und dem Nachweis ihrer  $\text{BIP}_{\mathbb{R}_+}$ -Berechenbarkeit. Ein zusätzlicher Abschnitt erläutert zuvor eine Reihe benötigter mathematischer Hilfsmittel.

#### 3.3.1 Ein $\text{PAR}_{\mathbb{R}_+}$ -vollständiges Entscheidungsproblem

Im Folgenden wird für die reelle Komplexitätsklasse  $\text{PAR}_{\mathbb{R}_+}$  das vollständige Entscheidungsproblem (HDTRAO, HDTRAO<sub>+</sub>) vorgestellt. Dabei handelt es sich um

### 3 Reelle interaktive Beweissysteme

eine eingeschränkte Variante des Erfüllbarkeitsproblems (TRAO, TRAO<sub>+</sub>) für Formeln in Logik erster Stufe über der Struktur der reellen Zahlen mit Addition und Ordnungsrelation.

Zunächst wird im Folgenden für die auf so genannte digitale Quantoren eingeschränkte Variante (DTRAO, DTRAO<sub>+</sub>) die PAR<sub>ℝ<sub>+</sub></sub>-Vollständigkeit bewiesen. Die nachfolgend angegebene Reduktion auf die zusätzlich konstantenfreie Variante (HDTRAO, HDTRAO<sub>+</sub>) schließt den Vollständigkeitsbeweis ab.

**Definition 3.3.1.** Sei TRAO (*theory of reals with addition and order*) die Menge aller vollständig quantifizierten booleschen Formeln  $\psi$ , konstruiert aus Variablen und reellen Konstanten mittels Additions-, Subtraktions- und Vergleichsoperationen. Sei weiter TRAO<sub>+</sub> die Menge aller Formeln  $\psi \in \text{TRAO}$ , die über  $\mathbb{R}$  wahr sind. Allgemein hat eine Formel  $\psi$  für  $Q_i \in \{\exists, \forall\}$  die Form

$$\psi = Q_1 x_1 \in \mathbb{R} \cdots Q_m x_m \in \mathbb{R}: \phi(x_1, \dots, x_m).$$

Dabei sei  $\phi$  eine quantorenfreie boolesche Formel mit Literalen der Form  $l = 0$  oder  $l > 0$ , wobei  $l$  eine lineare Funktion der folgenden Form ist

$$l(x_1, \dots, x_m) = \sum a_i x_i + \sum b_i c_i.$$

Die Koeffizienten  $a_i \in \mathbb{Z}$  der Variablen  $x_i$  sowie die Koeffizienten  $b_i \in \mathbb{Z}$  der reellen Konstanten  $c_i \in \mathbb{R}$  entstehen durch Additions- und Subtraktionsoperationen. Sie sind daher in ihrem Wert durch die Anzahl dieser Operationen innerhalb der Formel  $\psi$  beschränkt. Für eine entsprechende Kodierung von TRAO-Formeln gilt daher für den Wert der Koeffizienten die Abschätzung  $a_i \leq \text{size}(\psi)$  und  $b_i \leq \text{size}(\psi)$ .

**Definition 3.3.2.** Sei DTRAO (*digital theory of reals with addition and order*) eine Einschränkung der Menge TRAO derart, dass die quantifizierten Variablen  $x_1, \dots, x_m$  der Formeln  $\psi \in \text{DTRAO}$  der Nebenbedingung  $x_i = 0 \vee x_i = 1$  unterliegen. Sei entsprechend DTRAO<sub>+</sub> die Menge aller über  $\{0, 1\}$  wahren Formeln  $\psi \in \text{DTRAO}$ . In abkürzender Schreibweise haben die Formeln  $\psi \in \text{DTRAO}$  die Form

$$\psi = Q_1 x_1 \in \{0, 1\} \cdots Q_m x_m \in \{0, 1\}: \phi(x_1, \dots, x_m).$$

Dabei sei  $\phi$  weiterhin wie in Definition 3.3.1 eine ansonsten uneingeschränkte quantorenfreie boolesche Formel.

**Satz 3.3.3.** *Es gilt  $(\text{DTRAO}, \text{DTRAO}_+) \in \text{PAR}_{\mathbb{R}_+}$ .*

*Beweis.* Da alle quantifizierten Variablen von DTRAO-Formeln über der endlichen Teilmenge  $\{0, 1\} \subseteq \mathbb{R}$  quantifiziert sind, gibt es nur exponentiell viele mögliche Belegungen der quantifizierten Variablen. Die Enumeration dieser Belegungen und die Auswertungen der resultierenden unquantifizierten Formeln gelingt in paralleler polynomieller Zeit. Da in DTRAO-Formeln nur Additionen und Subtraktionen als arithmetische Operationen zulässig sind, können die Auswertungen von einer PAR<sub>ℝ<sub>+</sub></sub>-beschränkten Maschine durchgeführt werden.  $\square$

Um die Argumentation innerhalb des nachfolgend gemäß [CK94] wiedergegebenen  $\text{PAR}_{\mathbb{R}_+}$ -Vollständigkeitsbeweises für  $(\text{DTRAO}, \text{DTRAO}_+)$  zu erleichtern, wird als technisches Hilfsmittel zunächst eine eingeschränkte Variante der Komplexitätsklasse  $\text{PSPACE}_{\mathbb{R}_+}$  definiert. Der Vollständigkeitsbeweis wird für diese Klasse geführt, die – wie sich im Folgenden zeigt – identisch mit  $\text{PAR}_{\mathbb{R}_+}$  ist.

**Definition 3.3.4.** Sei für  $L \subseteq Y \subseteq \mathbb{R}^\infty$  die Klasse  $\mathcal{E}$  definiert als die Menge genau der Entscheidungsprobleme  $(Y, L)$ , für die eine additive BSS-Maschine  $M$  über  $Y$  und Konstanten  $c, k \in \mathbb{N}$  existieren, so dass  $(Y, L)$  von  $M$  entschieden wird und gilt

- (i)  $\forall y \in Y: T_M(y) \leq 2^{|y|^{k+c}} \wedge S_M(y) \leq |y|^k + c$  (also  $\mathcal{E} \subseteq \text{PSPACE}_{\mathbb{R}_+}$ )
- (ii) zu jedem Zeitpunkt der Berechnung entsprechen die Registerinhalte einer Linearkombination der Eingabewerte und der Maschinenkonstanten, wobei der Betrag der Koeffizienten kleiner als  $|y|^k + c$  ist.

**Lemma 3.3.5.** *Es gilt  $\text{PAR}_{\mathbb{R}_+} \subseteq \mathcal{E}$ .*

*Beweis.* Die in [Cuc93] verwendete Beweistechnik für Satz 1.4.16 führt unmittelbar auch zu einem Beweis für  $\text{PAR}_{\mathbb{R}_+} \subseteq \text{PSPACE}_{\mathbb{R}_+}$ . Die Zwischenergebnisse und die Ausgabe eines additiven algebraischen Schaltkreises polynomieller Tiefe sind bereits Linearkombinationen der geforderten Form. Diese Eigenschaft überträgt sich auch auf die simulierende  $\text{PSPACE}_{\mathbb{R}_+}$ -Maschine.  $\square$

**Satz 3.3.6.**  $(\text{DTRAO}, \text{DTRAO}_+)$  ist  $\mathcal{E}$ -vollständig bezüglich  $\leq_{\mathbb{R}_+}$ .

*Beweis.* Nach Satz 3.3.3 und Lemma 3.3.5 gilt  $(\text{DTRAO}, \text{DTRAO}_+) \in \mathcal{E}$ . Es genügt daher zu zeigen, dass  $(\text{DTRAO}, \text{DTRAO}_+)$   $\mathcal{E}$ -schwer ist bezüglich  $\leq_{\mathbb{R}_+}$ . Die im Folgenden verwendete Beweistechnik wurde bereits bei der Skizzierung des Beweises der  $\text{PSPACE}$ -Vollständigkeit von Q3SAT vorgestellt (siehe Satz 2.4.1).

Sei  $(Y, L) \in \mathcal{E}$ . Nach Definition 3.3.4 existiert eine  $\text{PSPACE}_{\mathbb{R}_+}$ -Maschine  $M$  mit Maschinenkonstanten  $c_1, \dots, c_r$ , die  $L$  über  $Y$  entscheidet. Zu  $M$  gibt es Konstanten  $k, c \in \mathbb{N}$ , so dass  $M$  für eine beliebige Eingabe  $y$  höchstens  $s_{\max} = |y|^k + c$  Register benutzt und nach höchstens  $t_{\max} = 2^{|y|^{k+c}}$  Zeitschritten terminiert.

Zu jedem Zeitschritt ist die Konfiguration von  $M$  gemäß Definition 1.1.3 durch ein Tupel  $(n, d, s, x_1, \dots, x_{s_{\max}})$  gegeben. Die Nummer der nächsten Instruktion  $n \in \mathbb{N}$  ist konstant beschränkt durch die Gesamtzahl der Instruktionen, die Registerindizes  $d \in \mathbb{N}$  und  $s \in \mathbb{N}$  sind durch die Laufzeit der Maschine polynomiell beschränkt. Wegen  $(Y, L) \in \mathcal{E}$  sind sämtliche Registerinhalte  $x_i$  Linearkombinationen der Eingabewerte und Maschinenkonstanten mit polynomiell beschränkten Koeffizienten.

Jede derartige Konfiguration  $(n, d, s, x_1, \dots, x_{s_{\max}})$  kann daher relativ zur Eingabe  $y$  und den Maschinenkonstanten  $c_1, \dots, c_r$  als Wort  $\alpha \in \{0, 1\}^*$  polynomieller Länge kodiert werden.

### 3 Reelle interaktive Beweissysteme

Für die Maschine  $M$ , eine Eingabe  $y \in Y$  und beliebige Konfigurationen  $\alpha$  und  $\beta$  können nun die folgenden Aussagen als DTRAO-Formeln angegeben werden:

$$\begin{aligned} \psi_{\text{start}}(\alpha, y) \text{ ist WAHR} &\Leftrightarrow \alpha \text{ ist eine Startkonfiguration für die Eingabe } y \\ \psi_{\text{akzept}}(\beta) \text{ ist WAHR} &\Leftrightarrow \beta \text{ ist eine akzeptierende Endkonfiguration} \\ \psi_{\text{gleich}}(\alpha, \beta) \text{ ist WAHR} &\Leftrightarrow \text{die Konfigurationen } \alpha \text{ und } \beta \text{ sind gleich} \\ \psi_{\text{folge}}(\alpha, \beta) \text{ ist WAHR} &\Leftrightarrow \beta \text{ ist bezüglich } M \text{ eine Folgekonfiguration von } \alpha \end{aligned}$$

Alle diese Formeln können in polynomieller Zeit ausgegeben werden, da die Länge der kodierten Konfigurationen  $\alpha, \beta$  polynomiell beschränkt ist. Des Weiteren beschreibt die im Folgenden für beliebige Konfigurationen  $\alpha$  und  $\beta$  induktiv definierte Formel  $\psi_m(\alpha, \beta)$  die Aussage, dass  $\beta$  bezüglich  $M$  in höchstens  $2^m$  Konfigurationsübergängen von  $\alpha$  aus erreichbar ist.

$$\begin{aligned} \psi_0(\alpha, \beta) &:= \psi_{\text{gleich}}(\alpha, \beta) \vee \psi_{\text{folge}}(\alpha, \beta) \\ \psi_{m+1}(\alpha, \beta) &:= \exists \gamma \forall \alpha' \forall \beta' : \left[ (\psi_{\text{gleich}}(\alpha', \alpha) \wedge \psi_{\text{gleich}}(\beta', \gamma)) \vee \right. \\ &\quad \left. (\psi_{\text{gleich}}(\alpha', \gamma) \wedge \psi_{\text{gleich}}(\beta', \beta)) \right] \Rightarrow \psi_m(\alpha', \beta') \end{aligned}$$

Die Länge der rekursiv expandierten Formel  $\psi_m(\alpha, \beta)$  ist linear von  $m$  abhängig und die Formel selbst kann in polynomieller Zeit bezüglich  $m$  ausgegeben werden.

Da  $y \in L$  genau dann gilt, wenn die Maschine  $M$  in höchstens  $t_{\max}$  Zeitschritten ausgehend von der Startkonfiguration eine akzeptierende Endkonfiguration erreicht, ist die gewünschte Reduktion gegeben durch  $y \mapsto \psi(y)$ . Mit  $t_{\max} = 2^{p(|y|)}$  folgt

$$\psi(y) := \exists \alpha \exists \beta : \psi_{\text{start}}(\alpha) \wedge \psi_{p(|y|)}(\alpha, \beta) \wedge \psi_{\text{akzept}}(\beta).$$

Durch einfache Umformungen (siehe Satz 2.4.1) kann  $\psi(y)$  in die allgemeine Form gemäß Definition 3.3.2 gebracht werden. Insgesamt kann die resultierende DTRAO-Formel in polynomieller Zeit konstruiert werden.  $\square$

**Korollar 3.3.7.**  $(\text{DTRAO}, \text{DTRAO}_+)$  ist  $\text{PAR}_{\mathbb{R}_+}$ -vollständig bezüglich  $\leq_{\mathbb{R}_+}$ .

*Beweis.*  $(\text{DTRAO}, \text{DTRAO}_+)$  ist  $\mathcal{E}$ -vollständig bezüglich  $\leq_{\mathbb{R}_+}$  und es gilt  $\text{PAR}_{\mathbb{R}_+} \subseteq \mathcal{E}$  sowie  $(\text{DTRAO}, \text{DTRAO}_+) \in \text{PAR}_{\mathbb{R}_+}$ .  $\square$

Ein wichtiger vorbereitender Schritt für die Reduktion von  $(\text{DTRAO}, \text{DTRAO}_+)$  auf ein diskretes Entscheidungsproblem ist die Elimination der reellen Konstanten aus den DTRAO-Formeln. Diese werden wie folgt durch freie Variablen ersetzt und es ergibt sich eine erfüllbarkeitsäquivalente homogene Darstellung.

**Definition 3.3.8.** Sei HDTRAO (*homogeneous digital theory of reals with addition and order*) die Menge aller Paare  $(\psi, c)$ , wobei  $c \in \mathbb{R}^n$  ein Vektor reeller Konstanten ist sowie  $\psi$  eine homogene (d. h. konstantenfreie) quantifizierte boolesche Formel,

konstruiert aus  $m \in \mathbb{N}$  quantifizierten und  $n$  freien Variablen mittels Additions-, Subtraktions- und Vergleichsoperationen. Die quantifizierten Variablen  $x_1, \dots, x_m$  unterliegen dabei der Einschränkung  $x_i = 0 \vee x_i = 1$ , bzw. in konstantenfreier Notation  $x_i = 0 \vee x_i = y_1$ , wobei o. B. d. A. für die freie Variable  $y_1 = 1$  gelte. Sei weiter  $\text{HDTRAO}_+$  die Menge aller Paare  $(\psi, c) \in \text{HDTRAO}$  derart, dass  $\psi(c)$  wahr ist. In abkürzender Schreibweise haben die Formeln  $\psi$  für  $\mathcal{Q}_i \in \{\exists, \forall\}$  die Form

$$\psi(y_1, \dots, y_n) = \mathcal{Q}_1 x_1 \in \{0, y_1\} \cdots \mathcal{Q}_m x_m \in \{0, y_1\} : \phi(x_1, \dots, x_m, y_1, \dots, y_n).$$

Dabei sei  $\phi$  eine quantorenfreie boolesche Formel mit Literalen der Form  $l = 0$  oder  $l > 0$ , wobei  $l$  eine konstantenfreie lineare Funktion der folgenden Form ist.

$$l(x_1, \dots, x_m, y_1, \dots, y_n) = \sum a_i x_i + \sum b_i y_i$$

Die Koeffizienten  $a_i \in \mathbb{Z}$  der Variablen  $x_i$  sowie die Koeffizienten  $b_i \in \mathbb{Z}$  der Variablen  $y_i \in \mathbb{R}$  entstehen durch Additions- und Subtraktionsoperationen und sind daher in ihrem Wert durch die Anzahl dieser Operationen innerhalb der Formel  $\psi$  beschränkt. Für eine entsprechenden Kodierung von HDTRAO-Formeln gilt daher für den Wert der Koeffizienten die Abschätzung  $a_i \leq \text{size}(\psi)$  und  $b_i \leq \text{size}(\psi)$ .

**Korollar 3.3.9.**  $(\text{HDTRAO}, \text{HDTRAO}_+)$  ist  $\text{PAR}_{\mathbb{R}_+}$ -vollständig.

*Beweis.* Aus einer HDTRAO-Formel  $(\psi', c)$  entsteht mittels Substitution der freien Variablen  $y_i$  durch die Konstanten  $c_i$  eine erfüllbarkeitsäquivalente DTRAO-Formel. Es gilt also  $(\text{HDTRAO}, \text{HDTRAO}_+) \in \text{PAR}_{\mathbb{R}_+}$ . Für den Nachweis der Vollständigkeit ist abschließend  $(\text{DTRAO}, \text{DTRAO}_+) \leq_{\mathbb{R}_+} (\text{HDTRAO}, \text{HDTRAO}_+)$  zu zeigen.

Sei  $\psi$  eine DTRAO-Formel mit den reellen Konstanten  $c_1, \dots, c_n$ . Dann sei  $\psi'$  die HDTRAO-Formel, die aus  $\psi$  entsteht, indem jedes Vorkommen einer Konstante  $c_i$  durch eine freie Variable  $y_i$  ersetzt wird. Insbesondere sei  $c_1 = 1$  durch  $y_1$  ersetzt. Für die resultierende Formel  $\psi'$  und die reellen Konstanten  $c_i$  gilt

$$\psi = \text{WAHR} \quad \Leftrightarrow \quad \psi'(c) = \text{WAHR}.$$

Die gesuchte Reduktion ist also gegeben durch die Abbildung  $\psi \mapsto (\psi', c)$ .  $\square$

### 3.3.2 Mathematische Hilfsmittel

Zur Vorbereitung des im nachfolgenden Abschnitt geführten Beweises der Reduzierbarkeit von  $(\text{HDTRAO}, \text{HDTRAO}_+)$  auf das diskrete Entscheidungsproblem Q3SAT werden im Folgenden einige mathematische Hilfsmittel in abgekürzter Darstellung eingeführt.

#### Effizientes Lösen linearer Gleichungssysteme im additiven BSS-Modell

Im additiven BSS-Modell können trotz des Fehlens einer allgemeinen Multiplikationsoperation binär kodierte ganze Zahlen miteinander multipliziert werden. Die

### 3 Reelle interaktive Beweissysteme

dazu geeigneten polynomiellen Verfahren entsprechen jenen, die auch auf klassischen Turingmaschinen zum Einsatz kommen.

Unter Verwendung eines elementaren Multiplikationsalgorithmus gelingt es darüber hinaus auch Multiplikationen von binär kodierten ganzen Zahlen mit reellen Zahlen zu berechnen. Der Aufwand dafür ist gemessen in der Anzahl benötigter reeller Additionen ebenfalls polynomiell.

Basierend auf diesen Grundoperationen können Lösungsverfahren für lineare Gleichungssysteme mit binär kodierten ganzzahligen Koeffizienten und reellem inhomogenen Anteil im additiven BSS-Modell implementiert werden. Ein entsprechender z. B. auf dem Gauss-Verfahren basierender Algorithmus hat nach [Sch86] eine polynomielle Laufzeit sowie polynomiellen Platzbedarf für die binär kodierten Zwischen- und Endergebnisse.

Sei also  $A \in \mathbb{Z}^{m \times n}$  eine Matrix binär kodiert vorliegender ganzer Zahlen und  $b \in \mathbb{R}^m$  der reelle inhomogene Anteil im linearen Gleichungssystem  $Ax = b$ . Da  $A$  ausschließlich ganzzahlige Koeffizienten hat, ist jeder Lösungsvektor  $x \in \mathbb{R}^n$ , der das Gleichungssystem erfüllt, rational von  $b$  abhängig. Mit Hilfe der zuvor beschriebenen Rechenoperationen können Lösungsvektoren in der Darstellungsform  $x = \frac{x'}{q}$  bestimmt werden, wobei  $x' \in \mathbb{R}^n$  und  $q \in \mathbb{Z}$ .

Eine additive BSS-Maschine kann daher bei der Eingabe einer Familie  $\{v_i\}_{i \in I}$  von binär kodierten ganzzahligen Vektoren  $v_i \in \mathbb{Z}^n$  und eines reellen Vektors  $x \in \mathbb{R}^n$  effizient folgende als lineare Gleichungssysteme formulierbare Probleme lösen.

1. Prüfe, ob die Familie von Vektoren  $\{v_i\}_{i \in I}$  linear unabhängig ist, falls ja:
2. Prüfe, ob der Vektor  $x$  eine Linearkombination von  $\{v_i\}_{i \in I}$  ist, falls ja:
3. Berechne die zugehörigen eindeutigen Koeffizienten der Form  $\lambda_i = \frac{\lambda'_i}{q_i}$ , wobei  $\lambda'_i \in \mathbb{R}$  und  $q_i \in \mathbb{Z}$ .

Die effiziente Lösbarkeit dieser drei Probleme ist eine Voraussetzung für den später geführten Reduktionsbeweis.

#### **Polyedrische Kegel**

Die nachfolgend definierten polyedrischen Kegel stellen ein geeignetes Abstraktionsmittel dar, um gewisse Eigenschaften der gesuchten Reduktion auf ein diskretes Entscheidungsproblem mathematisch zu beschreiben.

Insbesondere wird in dem später geführten Reduktionsbeweis der ohne Beweis [IdR98, Proposition 1] als Lemma 3.3.13 angeführte Spezialfall des Minkowski Theorems [SW70, Theorem (2.8.6)] benötigt.

**Definition 3.3.10.** Eine Menge  $C \subseteq \mathbb{R}^n$  ist ein *polyedrischer Kegel* genau dann, wenn sie Lösungsmenge eines endlichen Systems homogener linearer Ungleichungen ist, d. h.  $C = \{x \in \mathbb{R}^n \mid \forall i \in I: L_i(x) \leq 0\}$ , wobei  $\{L_i\}_{i \in I}$  ein endliches System homogener linearer Funktionen  $L_i: \mathbb{R}^n \rightarrow \mathbb{R}$  ist.

Weiter heißt ein polyedrischer Kegel  $C$  *geradenfrei* genau dann, wenn er keine Gerade enthält. Diese Bedingung ist äquivalent dazu, dass der Schnitt der Kerne aller linearen Abbildungen  $L_i$  nur den Ursprungspunkt enthält, in Zeichen  $\bigcap \ker L_i = \{0\}$ .

**Definition 3.3.11.** Für einen Vektor  $v \in \mathbb{R}^n$ ,  $v \neq 0$  ist der *Strahl*  $R(v)$  definiert als die Menge  $R(v) = \{\lambda v \mid \lambda \geq 0\}$ .

Weiter heißt ein Strahl  $R(v)$  *Kante eines polyedrischen Kegels*  $C$  genau dann, wenn  $v \in C$  gilt und  $v$  eine Lösung eines homogenen linearen  $(n-1) \times n$  Gleichungssystem vollen Rangs der Form  $\{L_{i_1}x = 0, \dots, L_{i_{n-1}}x = 0\}$  ist, wobei  $i_j \in I$ .

**Definition 3.3.12.** Die *Minkowski Summe von Mengen*  $X, Y \subseteq \mathbb{R}^n$ , in Zeichen  $X + Y$ , ist definiert als die Menge  $X + Y = \{x + y \mid x \in X \wedge y \in Y\}$ .

**Lemma 3.3.13.** Jeder geradenfreie polyedrische Kegel  $C \subseteq \mathbb{R}^n$  ist die Minkowski Summe seiner Kanten. Es gilt also

$$C = \{0\} + R(v_1) + \dots + R(v_N) = \left\{ \sum \lambda_i v_i \mid \lambda_i \geq 0 \right\},$$

wobei die  $v_i$  Vektoren sind, welche die Kanten von  $C$  erzeugen. □

### Satz von Carathéodory

Gemäß dem vorigen Lemma ist jeder polyedrische Kegel gleich der Minkowski Summe seiner Kanten. Entsprechend können alle Punkte eines polyedrischen Kegels als so genannte positive Linearkombinationen seiner Kantenvektoren aufgefasst werden.

Das folgende Lemma nach [SW70, Theorem (2.2.11)] stellt sicher, dass jeder einzelne Punkt des Kegels als positive Linearkombination einer linear unabhängigen Auswahl von Kantenvektoren darstellbar ist.

**Definition 3.3.14.** Ein Vektor  $x \in \mathbb{R}^n$  ist eine *positive Linearkombination einer Familie von Vektoren*  $\{v_i\}_{i \in I}$  genau dann, wenn es reelle Koeffizienten  $\lambda_i \geq 0$ ,  $i \in I$  gibt, so dass  $x = \sum_{i \in I} \lambda_i v_i$  gilt.

**Lemma 3.3.15 (Carathéodory).** Wenn ein Vektor  $x \in \mathbb{R}^n$  eine positive Linearkombination einer Familie von Vektoren  $\{v_i\}_{i \in I}$  ist, dann enthält  $\{v_i\}_{i \in I}$  eine linear unabhängige Unterfamilie  $\{v_i\}_{i \in J}$ , so dass  $x$  positive Linearkombination von  $\{v_i\}_{i \in J}$  ist. □

### Quantorenelimination mit dem Verfahren von Tarski

Nachfolgend wird kurz das klassische Resultat von Tarski [Tar51, Sei54] zur Quantorenelimination über dem reellen Zahlkörper vorgestellt. Dieses Resultat stellt im später geführten Reduktionsbeweis die Existenz einer quantorenfreien Darstellung von HDTRAO-Formeln sicher.

In diesem Kontext ist entscheidend, dass einige Eigenschaften der ursprünglichen Formel auch in der quantorenfreien Darstellung erhalten bleiben. Die Stabilität des zugehörigen algorithmischen Verfahrens bezüglich dieser nachfolgend aufgeführten Eigenschaften wird daher kurz begründet.

**Satz 3.3.16** (Tarski, Modelltheoretische Interpretation). *Der geordnete reelle Zahlkörper erlaubt in seiner zugehörigen Logik erster Stufe effektive Quantorenelimination. Dies bedeutet:*

Sei  $\psi(y_1, \dots, y_n)$  eine Formel in Logik erster Stufe über dem Körper  $\mathbb{R}$

$$\psi(y_1, \dots, y_n) = \mathbf{Q}_1 x_1 \cdots \mathbf{Q}_m x_m : \phi(x_1, \dots, x_m, y_1, \dots, y_n),$$

mit Quantoren  $\mathbf{Q}_i \in \{\exists, \forall\}$ , freien Variablen  $y_i \in \mathbb{R}$  und der quantorenfreien Formel  $\phi(x_1, \dots, x_m, y_1, \dots, y_n)$  als eine boolesche Verknüpfung von Polynomgleichungen und Ungleichungen der Form  $p(x_1, \dots, x_m, y_1, \dots, y_n) = 0$  bzw.  $p(x_1, \dots, x_m, y_1, \dots, y_n) > 0$ .

Dann kann effektiv, d. h. von einer BSS-Maschine berechenbar, eine erfüllbarkeitsäquivalente aber quantorenfreie Darstellung  $\phi'(y_1, \dots, y_n)$  von  $\psi(y_1, \dots, y_n)$  bestimmt werden:

$$\forall y \in \mathbb{R}^n : \psi(y) \Leftrightarrow \phi'(y). \quad \square$$

Anstelle eines Beweises wird im Folgenden eine stark vereinfachte Skizze des zugehörigen Verfahrens zur Quantorenelimination angegeben (vgl. [BCR87, MÖ04]).

Sei o. B. d. A. der quantorenfreie Teil  $\phi(x, y)$  der Formel  $\psi(y) = \mathbf{Q}_1 x_1 \cdots \mathbf{Q}_m x_m : \phi(x, y)$  eine Konjunktion von Polynomgleichungen und Ungleichungen der Form

$$\phi(x, y) = \bigwedge_{i=1}^k p_i(x, y) = 0 \quad \wedge \quad \bigwedge_{j=k+1}^{\ell} p_j(x, y) > 0$$

Nun sei  $P = \{p_i\}_{1 \leq i \leq \ell}$  das zugehörige Polynomsystem und  $\text{VZV}(P, x, y)$  der *Vorzeichenvektor* von  $P$  unter der Belegung  $(x, y)$  definiert durch

$$\text{VZV}(P, x, y) = (\text{sgn } p_1(x, y), \dots, \text{sgn } p_{\ell}(x, y)).$$

Dabei sei die Signumfunktion  $\text{sgn}: \mathbb{R} \rightarrow \{-1, 0, 1\}$  definiert durch

$$\text{sgn } x = \begin{cases} -1 & \text{falls } x < 0 \\ 0 & \text{falls } x = 0 \\ 1 & \text{falls } x > 0 \end{cases}.$$

Entsprechend gilt für die Erfüllbarkeit der quantifizierten Formel  $\psi(y)$

$$\forall y \in \mathbb{R}^n: \quad \psi(y) \Leftrightarrow \mathcal{Q}_1 x_1 \cdots \mathcal{Q}_m x_m: \text{VZV}(P, x, y) = \underbrace{(0, \dots, 0)}_k, \underbrace{1, \dots, 1)}_\ell$$

Ziel ist es also, die Realisierbarkeit des Vorzeichenvektors  $w_1 = (0, \dots, 0, 1, \dots, 1)$  im Polynomsystem  $P_1 = P$  unter der gegebenen Quantifizierung der Variablen  $x_1, \dots, x_m$  zu überprüfen. In einem rekursiven Verfahren werden dazu schrittweise die einzelnen quantifizierten Variablen eliminiert.

Entscheidend hierbei ist, dass die Quantifizierung einer Variable  $x_i$  über der überabzählbaren Grundmenge  $\mathbb{R}$  durch eine Quantifizierung über der endlichen Menge aller potentiell möglichen Vorzeichenvektoren  $w_{i+1}$  des Polynomsystems  $\text{BC}(P_i)$  ersetzt werden kann. Dabei ist  $\text{BC}(P_i)$  ein Polynomsystem über  $x_{i+1}, \dots, x_m$ , welches aus  $P_i$  durch Anwendung eines speziellen Hüllenoperators (*base closure*) entsteht.

Nach Tarski kann aus  $\text{BC}(P_i)$  und einem zugehörigen Vorzeichenvektor  $w_{i+1}$  das Vorzeichenmuster  $\text{VZM}(\text{BC}(P_i), w_{i+1})$ , d. h. die Folge von Vorzeichenvektoren berechnet werden, die  $P_i$  für  $x_i \in \mathbb{R}$  annehmen würde. Wird nun ein Vorzeichenvektor  $w_{i+1}$  gefunden derart, dass  $\text{VZM}(\text{BC}(P_i), w_{i+1})$  den ursprünglichen Vorzeichenvektor  $w_i$  enthält, so kann die Variable  $x_i$  als eliminiert betrachtet und im nächsten Rekursionsschritt die Realisierbarkeit von  $w_{i+1}$  im Polynomsystem  $P_{i+1} = \text{BC}(P_i)$  überprüft werden.

Im Folgenden wird beispielhaft der Rekursionsschritt von  $w_i$  zu  $w_{i+1}$  betrachtet, wenn  $\mathcal{Q}_i$  ein Existenzquantor ist. Der hier nicht berücksichtigte Fall eines Allquantors ist etwas komplizierter, kann aber im Wesentlichen analog betrachtet werden.

$$\begin{aligned} \exists x_i \mathcal{Q}_{i+1} x_{i+1} \cdots \mathcal{Q}_m x_m: \text{VZV}(P_i, x_i, x_{i+1}, \dots, x_m, y) = w_i &\Leftrightarrow \\ \exists \text{Vorzeichenvektor } w_{i+1}: \left[ \text{VZM}(\text{BC}(P_i), w_{i+1}) \text{ enthält } w_i \wedge \right. \\ \left. \mathcal{Q}_{i+1} x_{i+1} \cdots \mathcal{Q}_m x_m: \text{VZV}(\underbrace{\text{BC}(P_i)}_{=P_{i+1}}, x_{i+1}, \dots, x_m, y) = w_{i+1} \right] \end{aligned}$$

Der Rekursionsabstiegs endet mit einem Vorzeichenvektor  $w_{m+1}$ , dessen Realisierbarkeit trivial überprüft werden kann. Es ergibt sich folgende erfüllbarkeitsäquivalente aber quantorenfreie Darstellung von  $\psi(y)$

$$\phi'(y) = \left[ \text{VZV}(\text{BC}(P_m), y) = w_{m+1} \right].$$

*Bemerkung 3.3.17.* Die unquantifizierte Formel  $\phi'$  entsteht aus der unquantifizierten Teilformel  $\phi$ , indem auf das zugehörige Polynomsystem  $P_1$  wiederholt der BC-Hüllenoperator angewandt wird.

Entsprechend werden die Eigenschaften der Formel  $\phi'$  durch die Eigenschaften des Hüllenoperators bestimmt. Im Wesentlichen entsteht  $\text{BC}(P_i)$ , indem  $P_i$  um die Ableitungen nach  $x_i$  sowie um die Reste von Polynomdivisionen seiner Polynome ergänzt wird. Folgende Eigenschaften der Teilformel  $\phi$  bleiben in  $\phi'$  erhalten:

### 3 Reelle interaktive Beweissysteme

1. der maximale Grad der Polynome bleibt erhalten,
2. die Größenordnung der Koeffizienten bleibt erhalten,
3. die reellen Konstanten bleiben unverändert erhalten.

Ein bedeutender Unterschied ist hingegen, dass die Formel  $\phi'$  bedingt durch das beträchtliche Anwachsen der Polynomsysteme  $P_i$  bei Anwendung der Hüllenoperation deutlich länger ist, als die Formel  $\phi'$ .

#### 3.3.3 Reduktion auf ein diskretes Entscheidungsproblem

Die zentrale Idee bei der Reduktion einer HDTRAO-Instanz  $(\psi, x)$  auf eine Instanz des diskreten Entscheidungsproblems Q3SAT besteht darin, den reellen Vektor  $x$  durch einen kleinen ganzzahligen Vektor  $v$  zu ersetzen. Dieser wird so gewählt, dass die Formel  $\psi$  genau dann durch  $v$  erfüllt wird, wenn dies auch für  $x$  gilt. Anschließend kann  $(\psi, v)$  in eine äquivalente Q3SAT-Instanz  $\psi'$  umgeformt werden.

In [Son85] wird erstmals eine geometrische Methode zur Abschätzung der Größe von erfüllbarkeitsäquivalenten ganzzahligen Vektoren in Abhängigkeit von verschiedenen Eigenschaften der Formel  $\psi$  und des reellen Vektors  $x$  angegeben.

Ivanov und de Rougemont [IdR98] geben ein  $h$ -Dekomposition genanntes Verfahren an, dass zu einer HDTRAO-Instanz  $(\psi, x)$  einen kleinen, erfüllbarkeitsäquivalenten und ganzzahligen Vektor  $v$  bestimmt. Dieses Verfahren wird nachfolgend vorgestellt und anschließend für die Konstruktion eines  $\text{BIP}_{\mathbb{R}^+}$ -Protokolls zur Reduktion von HDTRAO auf das diskrete Entscheidungsproblem Q3SAT genutzt.

**Definition 3.3.18.** Für ein  $h \in \mathbb{N}$  und einen reellen Vektor  $x \in \mathbb{R}^n$  heißt eine endliche Familie  $\{v_1, \dots, v_k\}$  ganzzahliger Vektoren  $v_i \in \mathbb{Z}^n$  genau dann  $h$ -Dekomposition von  $x$ , wenn folgende Bedingungen erfüllt sind:

- (i) für die Komponenten aller Vektoren  $v_i = (v_{i,1}, \dots, v_{i,n})$  gilt  $\text{bitsize}(v_{i,j}) \leq hn^2$ ,
- (ii)  $v_1, \dots, v_k$  sind linear unabhängig,
- (iii)  $x$  ist eine positive Linearkombination von  $\{v_1, \dots, v_k\}$  und
- (iv) für alle lineare Funktionen  $L: \mathbb{R}^n \rightarrow \mathbb{R}$  mit ganzzahligen Koeffizienten  $\leq h$  gilt

$$\forall v_i: L(v_i) \geq 0 \quad \vee \quad \forall v_i: L(v_i) \leq 0.$$

Die  $h$ -Dekomposition des Nullvektors ist die leere Familie von Vektoren.

Als  $h$ -Dekomposition eines Punkts  $x$  eignen sich die Kantenvektoren eines polyedrischen Kegels, der den Punkt  $x$  in einem gewissen, von  $h$  abhängigen Sinne dicht umschließt. Auf diese Weise wird erreicht, dass die den polyedrischen Kegel

konstituierenden Kantenvektoren stets gemeinsam ober- bzw. unterhalb der Null-ebene einer beliebigen linearen Funktion mit ganzzahligen durch  $h$  beschränkten Koeffizienten liegen (siehe Bedingung (iv) in Definition 3.3.18). Entsprechend gelten gewisse durch lineare Gleichungen und Ungleichungen ausdrückbare Eigenschaften des Punktes  $x$  auch für alle übrigen im Kegel enthaltene Punkte.

**Lemma 3.3.19.** *Für jedes  $h \in \mathbb{N}$  und jedes  $x \in \mathbb{R}^n$  existiert eine  $h$ -Dekomposition von  $x$ .*

*Beweis.* Zu einem  $x \in \mathbb{R}^n$  sei  $C_h(x)$  der wie folgt definierte polyedrische Kegel: Sei  $\{L_i\}$  die endliche Familie aller linearen Funktionen  $L_i: \mathbb{R}^n \rightarrow \mathbb{R}$  mit ganzzahligen Koeffizienten  $\leq h$ , für die  $x$  Lösung der zugehörigen homogenen linearen Ungleichung  $L_i(x) \leq 0$  ist. Dann ist  $C_h(x)$  die Lösungsmenge des resultierenden homogenen linearen Ungleichungssystems  $\{L_i \leq 0\}$ .

Der polyedrische Kegel  $C_h(x)$  ist geradenfrei, da das ihn definierende Ungleichungssystem  $\{L_i \leq 0\}$  für jedes  $j = 1, \dots, n$  mindestens eine der Ungleichungen  $x_j \leq 0$  oder  $-x_j \leq 0$  enthält. Entsprechend kann  $C_h(x)$  keine Gerade enthalten, deren Richtungsvektor in der  $j$ -ten Komponente von null verschieden ist.

Seien  $v_1, \dots, v_N \in \mathbb{Z}^n$  die Vektoren, welche die Kanten von  $C_h(x)$  erzeugen. Die  $v_i$  sind Lösungen von geeignet aus  $\{L_i\}$  gewählten homogenen linearen  $(n-1) \times n$  Gleichungssystemen mit ganzzahligen Koeffizienten  $\leq h$ . Durch eine geeignete Erweiterung eines dieser Gleichungssysteme auf ein  $n \times n$  System, dessen eindeutige Lösung  $v_i$  ist, kann der maximale Wert der Komponenten von  $v_i$  z. B. mit der Cramerschen Regel abgeschätzt werden. Demnach ist die Determinante der Koeffizientenmatrix eine obere Schranke, deren Wert wiederum mit Hilfe des Laplaceschen Entwicklungssatzes auf  $\leq n! h^n$  abgeschätzt werden kann. Für die Kodierungslänge der Komponenten von  $v_i = (v_{i,1}, \dots, v_{i,n})$  gilt entsprechend  $\text{bitsize}(v_{i,j}) \leq hn^2$ . Somit erfüllen die  $v_i$  die Bedingung (i) von Definition 3.3.18.

Nach Lemma 3.3.13 gilt  $C_h(x) = \{\sum \lambda_i v_i \mid \lambda_i \geq 0\}$ . Es gilt insbesondere  $x = \sum \lambda_i v_i$  für bestimmte  $\lambda_i \geq 0$ . Der Vektor  $x$  ist somit eine positive Linearkombination der Familie von Vektoren  $\{v_1, \dots, v_N\}$ . Aus dieser kann gemäß Lemma 3.3.15 eine linear unabhängige Unterfamilie  $\{v_i\}_{i \in J \subseteq \{1, \dots, N\}}$  gewählt werden, so dass  $x$  ebenfalls eine positive Linearkombination von dieser ist. Damit sind die Bedingungen (ii) und (iii) von Definition 3.3.18 erfüllt.

Die Bedingung (iv) der Definition wird von den  $v_i$  aufgrund ihrer Konstruktion und wegen  $v_i \in C_h(x)$  erfüllt.

Somit ist  $\{v_i\}_{i \in J}$  die gesuchte  $h$ -Dekomposition von  $x$ . □

**Beispiel 3.3.20.** Im Folgenden wird gemäß Lemma 3.3.19 eine  $h$ -Dekomposition in  $\mathbb{R}^2$  für  $h = 3$  und  $x = (e, \pi)$  konstruiert. Dazu wird zunächst die Menge  $\mathcal{L}$  aller

### 3 Reelle interaktive Beweissysteme

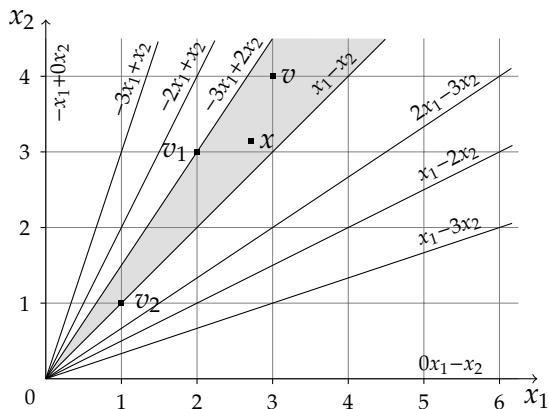


Abbildung 3.1:  $h$ -Dekomposition von  $x$  in  $\mathbb{R}^2$ , für  $h = 3$  und  $x = (e, \pi)$

homogenen linearen Funktionen  $L_i: \mathbb{R}^2 \rightarrow \mathbb{R}$  mit ganzzahligen Koeffizienten  $\leq 3$  betrachtet.

$$\mathcal{L} = \{ x_1, \underline{-x_1}, x_2, \underline{-x_2}, \\ x_1+x_2, x_1+2x_2, x_1+3x_2, 2x_1+x_2, 2x_1+3x_2, 3x_1+x_2, 3x_1+2x_2, \\ \underline{x_1-x_2}, \underline{x_1-2x_2}, \underline{x_1-3x_2}, 2x_1-x_2, \underline{2x_1-3x_2}, 3x_1-x_2, 3x_1-2x_2, \\ -x_1+x_2, -x_1+2x_2, -x_1+3x_2, \underline{-2x_1+x_2}, -2x_1+3x_2, \underline{-3x_1+x_2}, \underline{-3x_1+2x_2}, \\ \underline{-x_1-x_2}, \underline{-x_1-2x_2}, \underline{-x_1-3x_2}, \underline{-2x_1-x_2}, \underline{-2x_1-3x_2}, \underline{-3x_1-x_2}, \underline{-3x_1-2x_2} \}$$

Die in vorstehender Menge  $\mathcal{L}$  unterstrichen dargestellten linearen Funktionen  $\underline{L}_i$  erfüllen die Bedingung  $\underline{L}_i(x) \leq 0$ . Das zu diesen Funktionen gehörige homogene lineare Ungleichungssystem  $\{\underline{L}_i \leq 0\}$  definiert einen polyedrischen Kegel  $C$ , dessen Kanten durch zwei Vektoren  $v_1, v_2 \in C$  erzeugt werden, die jeweils Lösung einer homogenen linearen Gleichung  $\underline{L}_i = 0$  sind.

Abbildung 3.1 zeigt die im ersten Quadranten liegenden Nulllinien, d. h. die Kerne der linearen Funktionen  $\underline{L}_i$ . Der polyedrische Kegel  $C$  ist als grau hinterlegte Fläche dargestellt. Ferner ist  $v$  die Summe der kantenerzeugenden Vektoren  $v_1$  und  $v_2$ .

**Lemma 3.3.21.** Sei  $\psi$  eine HDTRAO-Formel mit  $n$  freien Variablen und  $\text{size}(\psi) = h$ . Seien weiter  $y, y' \in \mathbb{R}^n$  zwei reelle Vektoren derart, dass für alle linearen Funktionen  $L: \mathbb{R}^n \rightarrow \mathbb{R}$  mit ganzzahligen Koeffizienten  $\leq h$  die Vorzeichen von  $L(y)$  und  $L(y')$  übereinstimmen, d. h.  $\text{sgn } L(y) = \text{sgn } L(y')$ . Dann gilt auch  $\psi(y) \Leftrightarrow \psi(y')$ .

*Beweis.* Gemäß Satz 3.3.16 kann die quantifizierte Formel  $\psi(y)$  in eine quantorenfreie Darstellung  $\phi'(y)$  überführt werden. Wegen der in Bemerkung 3.3.17 genannten besonderen Eigenschaften des Eliminationsverfahrens sind die Literale von  $\phi'$  ebenso wie zuvor die Literale von  $\psi$  konstantenfreie homogene lineare Gleichungen und

Ungleichungen mit ganzzahligen Koeffizienten. Sie haben die Form  $L(y) = 0$  bzw.  $L(y) > 0$ . Der Wert der Koeffizienten von  $\phi'$  sowie von  $\psi$  ist durch die Länge der Formel  $\psi$  beschränkt und daher  $\leq h$ .

Der Wert von  $\phi'(y)$  bzw.  $\phi'(y')$  hängt ausschließlich von den Vorzeichen der Literale unter der Belegung  $y$  bzw.  $y'$  ab. Da nach Voraussetzung die Vorzeichen aller linearen Funktionen mit ganzzahligen Koeffizienten  $\leq h$  unter der Belegung  $y$  und  $y'$  jeweils übereinstimmen, gilt dies auch für die Literale von  $\phi'$ . Somit folgt die Behauptung.  $\square$

**Lemma 3.3.22.** Sei  $\psi$  eine HDTRAO-Formel mit  $\text{size}(\psi) = h$  und  $x \in \mathbb{R}^n$ . Weiter sei  $\{v_1, \dots, v_k\}$  eine  $h$ -Dekomposition von  $x$  und  $v = \sum v_i$ . Dann gilt  $\psi(x) \Leftrightarrow \psi(v)$ .

*Beweis.* Gemäß Lemma 3.3.21 genügt es zu zeigen, dass für alle linearen Funktionen  $L: \mathbb{R}^n \rightarrow \mathbb{R}$  mit ganzzahligen Koeffizienten  $\leq h$  die Vorzeichen von  $L(x)$  und  $L(v)$  übereinstimmen, d. h.  $\text{sgn } L(x) = \text{sgn } L(v)$ .

Da  $x = \sum \lambda_i v_i$  mit  $\lambda_i \geq 0$  und  $v = \sum \lambda'_i v_i$  mit  $\lambda'_i \geq 0$  (positive) Linearkombinationen von  $\{v_1, \dots, v_k\}$  sind, gilt definitionsgemäß für jede der linearen Funktionen  $L$

$$L(x) = \sum \lambda_i L(v_i) \quad \text{und} \quad L(v) = \sum \lambda'_i L(v_i).$$

Sei nun  $L(v_i) = 0$  für alle  $1 \leq i \leq k$ , dann gilt

$$L(x) = \sum \lambda_i L(v_i) = 0 = \sum \lambda'_i L(v_i) = L(v).$$

Andernfalls haben gemäß der Definition einer  $h$ -Dekomposition alle von null verschiedenen Werte  $L(v_i)$  das selbe Vorzeichen. Wegen  $\lambda_i, \lambda'_i \geq 0$  bleibt dieses Vorzeichen auch für  $L(x)$  und  $L(v)$  erhalten und es gilt

$$\text{sgn } L(x) = \text{sgn } \sum \lambda_i L(v_i) = \text{sgn } \sum \lambda'_i L(v_i) = \text{sgn } L(v). \quad \square$$

**Definition 3.3.23.** Eine Instanz  $(\psi, v) \in \text{HDTRAO}$  mit  $v \in \mathbb{Z}^n$  heißt *diskretisiert bezüglich*  $x \in \mathbb{R}^n$  genau dann, wenn

- (i)  $\psi(x) \Leftrightarrow \psi(v)$ ,
- (ii)  $\text{bitsize}(v)$  ist polynomiell bezüglich  $\text{size}(\psi)$  und  $n$ .

**Lemma 3.3.24.** Sei  $(\psi, x) \in \text{HDTRAO}$ ,  $x \in \mathbb{R}^n$ . Es existiert eine  $\text{BIP}_{\mathbb{R}_+}$ -berechenbare Abbildung  $(\psi, x) \mapsto (\psi, v)$  derart, dass  $(\psi, v)$  diskretisiert bezüglich  $x$  ist.

*Beweis.* Gemäß Lemma 3.3.22 genügt es zu zeigen, dass ein  $\text{BIP}_{\mathbb{R}_+}$ -Protokoll existiert, welches bei Eingabe von  $(\psi, x)$  mit  $\text{size}(\psi) = h$  eine  $h$ -Dekomposition  $\{v_1, \dots, v_k\}$  von  $x$ , bzw. den Summenvektor  $v = \sum v_i$  berechnet.

Offensichtlich bestimmt Algorithmus 3.1 für die Eingabe  $(\psi, x)$  korrekt die zugehörige  $h$ -Dekomposition: Nachdem der Beweiser eine potentielle  $h$ -Dekomposition

---

**Algorithmus 3.1** : BIP $_{\mathbb{R}_+}$ -Protokoll zur  $h$ -Dekomposition

---

**Eingabe** : HDTRAO-Instanz  $(\psi, x)$ , wobei  $h = \text{size}(\psi)$  und  $x \in \mathbb{R}^n$

**Ausgabe** : Summenvektor  $v \in \mathbb{Z}^n$  der  $h$ -Dekomposition von  $x$

empfangen  $k \in \mathbb{N}, k \leq n$  vom Beweiser

empfangen  $v_1, \dots, v_k \in \mathbb{Z}^n$  mit  $\text{bitsize}(v_{i,j}) \leq hn^2$  vom Beweiser (i)

prüfe, ob  $v_1, \dots, v_k$  linear unabhängig sind, sonst verwerfe (ii)

berechne  $\lambda_1, \dots, \lambda_k$ , so dass  $x = \sum_{i=1}^k \lambda_i v_i$ , falls diese nicht existieren, verwerfe  
 prüfe, ob  $\lambda_i \geq 0$  für alle  $i \in \{1, \dots, k\}$ , sonst verwerfe (iii)

prüfe, ob  $\forall i: L(v_i) \leq 0$  oder  $\forall i: L(v_i) \geq 0$  für alle lineare Funktionen  $L: \mathbb{R}^n \rightarrow \mathbb{R}$   
 mit ganzzahligen Koeffizienten  $\leq h$ , sonst verwerfe (iv)

berechne  $v := v_1 + \dots + v_k$

Ausgabe von  $v$

---

$\{v_1, \dots, v_k\}$  bereitgestellt hat, überprüft der Verifizierer der Reihe nach die in Definition 3.3.18 formulierten Bedingungen (i) bis (iv).

Es bleibt zu zeigen, dass diese Bedingungen innerhalb eines BIP $_{\mathbb{R}_+}$ -Protokolls geprüft werden können. Dies trifft trivialerweise für Bedingung (i) zu. Die Bedingungen (ii) und (iii) lassen sich als lineare Gleichungssysteme ausdrücken. Diese können gemäß den Ausführungen in Abschnitt 3.3.2 effizient von einer deterministischen, additiven BSS-Maschine gelöst werden.

Die Bedingung (iv) ist das Komplement der folgenden NP-Bedingung: Es existiert eine lineare Funktion  $L: \mathbb{R}^n \rightarrow \mathbb{R}$  mit ganzzahligen Koeffizienten  $\leq hn^2$  und  $i, j \in \{1, \dots, k\}$ , so dass  $L(v_i) < 0 \wedge L(v_j) > 0$ . Entsprechend ist Bedingung (iv) eine co-NP-Bedingung und kann daher gemäß Korollar 2.3.6 durch ein IP- und somit auch durch ein BIP $_{\mathbb{R}_+}$ -Protokoll geprüft werden.  $\square$

**Satz 3.3.25.** *Es gilt HDTRAO  $\in$  BIP $_{\mathbb{R}_+}$ .*

*Beweis.* Gemäß Lemma 3.3.24 und dem Resultat von Shamir (Satz 2.4.5) genügt es zu zeigen, dass die bezüglich der Eingabe  $(\psi, x)$  diskretisierte HDTRAO-Instanz  $(\psi, v)$  auf eine erfüllbarkeitsäquivalente Q3SAT-Instanz  $\psi'$  reduziert werden kann.

Die diskretisierte HDTRAO-Instanz  $(\psi, v)$  mit  $v = (v_1, \dots, v_n) \in \mathbb{Z}^n$  kann gemäß Definition 3.3.8 in folgender Form notiert werden

$$\psi(v_1, \dots, v_n) = Q_1 x_1 \in \{0, v_1\} \cdots Q_m x_m \in \{0, v_1\} : \phi(x_1, \dots, x_m, v_1, \dots, v_n).$$

Mit dem Ziel der Umformung in eine Q3SAT-Instanz wird zunächst die Quantifizierung der Variablen angepasst. Es ergibt sich

$$\psi(v_1, \dots, v_n) = Q_1 b_1 \in \{0, 1\} \cdots Q_m b_m \in \{0, 1\} : \phi(b_1 v_1, \dots, b_m v_1, v_1, \dots, v_n).$$

Die Literale der quantorenfreien Formel  $\phi$  sind konstantenfreie homogene lineare Gleichungen und Ungleichungen über den Produkten  $b_1v_1, \dots, b_mv_1$  und den Variablen  $v_1, \dots, v_n$ . Da die ganzzahligen Variablen  $v_1, \dots, v_n$  polynomiell in ihrem Wert beschränkt sind, können die zur Auswertung der linearen Gleichung und Ungleichungen nötigen Berechnungen jeweils als boolesche Formel polynomieller Länge kodiert werden. Dies gelingt analog zu der im Beweis des Satzes von Cook bzw. der  $\text{NP}_{\mathbb{R}}$ -Vollständigkeit von 2-SAS angewandten Methode.

Insgesamt ergibt sich eine quantifizierte boolesche Formel polynomieller Länge ohne freie Variablen. Für diese Formel  $\psi'$  gilt aufgrund ihrer Konstruktion

$$\psi(v) \Leftrightarrow \psi'.$$

Der im Folgenden angegebene  $\text{BIP}_{\mathbb{R}_+}$ -Verifizierer für  $(\text{HDTRAO}, \text{HDTRAO}_+)$  fasst die drei benötigten Teilschritte zusammen. Die akkumulierte Fehlerwahrscheinlichkeit der beiden interaktiven Beweissysteme im ersten und im dritten Teilschritt ist kleiner als  $\frac{1}{2} - \varepsilon$ . Sie ist für die positiven Instanzen  $(\psi, x) \in \text{HDTRAO}_+$  sogar gleich null. Entsprechend folgt die Behauptung.  $\square$

---

**Algorithmus 3.2 :**  $\text{BIP}_{\mathbb{R}_+}$ -Verifizierer  $V$  für  $(\text{HDTRAO}, \text{HDTRAO}_+)$

---

**Eingabe :** HDTRAO-Instanz  $(\psi, x)$ ,  $x \in \mathbb{R}^n$

**Ausgabe :** akzeptiere, falls  $(\psi, x) \in \text{HDTRAO}_+$ , sonst verwerfe

berechne eine Diskretisierung  $(\psi, v)$  bezüglich  $x$   $(\psi, x) \mapsto (\psi, v)$

bestimme eine erfüllbarkeitsäquivalente Q3SAT-Instanz  $\psi'$   $(\psi, v) \mapsto \psi'$

entscheide die Q3SAT-Instanz  $\psi'$

---

Wegen der  $\text{PAR}_{\mathbb{R}_+}$ -Vollständigkeit von  $(\text{HDTRAO}, \text{HDTRAO}_+)$  und der zuvor bewiesenen Inklusion  $\text{BIP}_{\mathbb{R}} \subseteq \text{PAR}_{\mathbb{R}}$  ergibt sich das folgende Korollar.

**Korollar 3.3.26.** *Es gilt  $\text{BIP}_{\mathbb{R}_+} = \text{PAR}_{\mathbb{R}_+}$ .*  $\square$

### 3.4 $\text{BIP}_{\mathbb{R}} \subseteq \text{IP}_{\mathbb{R}_+}$

In diesem Abschnitt werden die Separationsresultate von Ivanov und de Rougemont vorgestellt. Diese ergeben sich aus der besonderen Schwierigkeit zu entscheiden, ob eine gegebene große reelle Zahl, z. B. der Größenordnung  $2^{2^n}$ , eine ganze Zahl ist (vgl. [Mee95]). Im Folgenden wird für ein derartiges Problem gezeigt, dass es nicht in  $\text{PAR}_{\mathbb{R}}$  und daher nicht in  $\text{BIP}_{\mathbb{R}}$ , wohl aber in  $\text{IP}_{\mathbb{R}_+}$  enthalten ist.

### 3 Reelle interaktive Beweissysteme

Ein einfacheres Problem ist es, zu entscheiden, ob eine reelle Zahl  $x$  der Größenordnung  $2^n$  ganzzahlig ist. Das Problem kann auf die Frage  $x - [x] = 0$  reduziert werden, wobei  $[x] = \max\{z \in \mathbb{Z} \mid z \leq x\}$  der ganzzahlige Anteil von  $x$  sei.

Algorithmus 3.3 berechnet effizient den ganzzahligen Anteil  $[x]$  einer reellen Zahl  $x \in [0, 2^n]$ , indem er von  $x$  sukzessive die größten Zweierpotenzen  $\leq 2^n$  subtrahiert, so dass  $x$  größer als null bleibt. Die Summe dieser Zweierpotenzen ist der ganzzahlige Anteil von  $x$ . Die Folge der Zweierpotenzen ergibt sich durch fortgesetztes Addieren der letzten Potenz zu sich selbst, beginnend bei  $2^0 = 1$ . Dieses Verfahren benötigt auf einer additiven BSS-Maschine die Zeit  $O(n)$ .

---

**Algorithmus 3.3** :  $\mathbb{P}_{\mathbb{R}_+}$ -Algorithmus für  $x \mapsto [x]$

---

**Eingabe** : Eingabevektor  $(x, 1, \dots, 1) \in \mathbb{R}^{n+1}$

**Ausgabe** : Ganzzahliger Teil  $[x]$ , falls  $0 \leq x \leq 2^n$ , sonst verwerfe

**Daten** : Zweierpotenzen  $p_0 = 2^0, p_1 = 2^1, \dots, p_n = 2^n$

setze  $p_0 := 1$  und  $x' := 0$

**for**  $i := 1$  **to**  $n$  **do**

$\perp$  setze  $p_i := p_{i-1} + p_{i-1}$

prüfe, ob  $0 \leq x \leq p_n$ , sonst verwerfe

**for**  $i := n$  **downto**  $0$  **do**

$\perp$  **if**  $x - p_i \geq 0$  **then**

$\perp$  setze  $x := x - p_i$  und  $x' := x' + p_i$

Ausgabe von  $x'$

---

Die nächsten Abschnitte sind wie folgt gegliedert: Nachdem zunächst die Entscheidungsprobleme PROD und INT vorgestellt werden, folgt ein Beweis für  $\text{INT} \notin \text{PAR}_{\mathbb{R}}$ . Anschließend wird ein Beweis für  $\text{PROD} \in \text{IP}_{\mathbb{R}_+}$  und darauf basierend ein Beweis für  $\text{INT} \in \text{IP}_{\mathbb{R}_+}$  geführt. Als Folgerung ergibt sich die Separation der additiven sowie der multiplikativen Komplexitätsklassen:  $\text{BIP}_{\mathbb{R}_+} \subsetneq \text{IP}_{\mathbb{R}_+}$  und  $\text{BIP}_{\mathbb{R}} \subsetneq \text{IP}_{\mathbb{R}}$ . Abschließend wird als Anwendung dieser Resultate gezeigt, dass alle diskreten Entscheidungsprobleme in  $\text{IP}_{\mathbb{R}_+}$  enthalten sind.

#### 3.4.1 Zwei Ganzzahl-Entscheidungsprobleme

Es folgen die Definition der Entscheidungsprobleme PROD und INT.

**Definition 3.4.1.** Die Entscheidungsprobleme  $(\mathcal{P}, \text{PROD})$  und  $(\mathcal{I}, \text{INT})$  über den Mengen zulässiger Eingaben  $\mathcal{P} = \{(x, y, 1, \dots, 1) \in \mathbb{R}^{\infty}\}$  und  $\mathcal{I} = \{(x, 1, \dots, 1) \in \mathbb{R}^{\infty}\}$

seien durch ihre positiven Instanzen wie folgt definiert.

$$\begin{aligned} \text{PROD}_n &= \{ (x, y, 1, \dots, 1) \in \mathbb{R}^{n+2} \mid x = y \cdot 2^{2^n} \} & \text{PROD} &= \bigcup_{i \in \mathbb{N}} \text{PROD}_i \\ \text{INT}_n &= \{ (x, 1, \dots, 1) \in \mathbb{R}^{n+1} \mid x \in \mathbb{Z} \cap [0, 2^{2^n}] \} & \text{INT} &= \bigcup_{i \in \mathbb{N}} \text{INT}_i \end{aligned}$$

Gemäß folgendem Lemma liefert jedes Separationsresultat bezüglich  $(\mathcal{I}, \text{INT})$  auch eine negative obere Schranke für die Komplexität der Quantorenelimination über den reellen Zahlen mit Addition und Ordnung.

**Lemma 3.4.2.** *Es gilt  $(\mathcal{I}, \text{INT}) \leq_{\mathbb{R}_+} (\text{TRAO}, \text{TRAO}_+)$ .*

*Beweis.* Die Mengen  $\text{PROD}$  und  $\text{INT}$  können alternativ durch quantifizierte Formeln definiert werden, d. h. als Menge von reellen Vektoren, die diese Formeln erfüllen. Mit den entsprechenden Prädikaten  $\text{PROD}_n(x, y)$  und  $\text{INT}_n(x)$  folgt

$$\begin{aligned} \text{PROD}_n &= \{ (x, y, 1, \dots, 1) \in \mathbb{R}^{n+2} \mid \text{PROD}_n(x, y) \}, \\ \text{INT}_n &= \{ (x, 1, \dots, 1) \in \mathbb{R}^{n+1} \mid \text{INT}_n(x) \}. \end{aligned}$$

Offensichtlich gilt  $x = y \cdot 2^{2^n} \Leftrightarrow x = y \cdot 2^{2^{n-1}} \cdot 2^{2^{n-1}}$ . Entsprechend kann das Prädikat  $\text{PROD}_n(x, y)$  wie folgt rekursiv als quantifizierte Formel definiert werden.

$$\begin{aligned} \text{PROD}_0(x, y) &= (x=y+y) \\ \text{PROD}_n(x, y) &= \exists z: \text{PROD}_{n-1}(x, z) \wedge \text{PROD}_{n-1}(z, y) \end{aligned}$$

Des Weiteren kann jede reelle Zahl  $x \in \mathbb{Z} \cap [0, 2^{2^n}]$  zur Basis  $2^{2^{n-1}}$  dargestellt werden als  $x = y \cdot 2^{2^{n-1}} + z$  mit  $y, z \in \mathbb{Z} \cap [0, 2^{2^{n-1}}]$ . Das Prädikat  $\text{INT}_n(x)$  kann daher wie folgt rekursiv als quantifizierte Formel definiert werden.

$$\begin{aligned} \text{INT}_0(x) &= (x=0 \vee x=1 \vee x=2) \\ \text{INT}_n(x) &= \exists y, y', z: x=y'+z \wedge \text{PROD}_{n-1}(y', y) \wedge \text{INT}_{n-1}(y) \wedge \text{INT}_{n-1}(z) \end{aligned}$$

Durch Einführung zusätzlicher allquantifizierter Variablen können diese Formeln so abgewandelt werden, dass sich nach ihrer rekursiven Expansion jeweils Formeln polynomieller Länge für  $\text{PROD}_n$  und  $\text{INT}_n$  ergeben. Dazu ist beispielsweise  $\text{INT}_{n-1}(y) \wedge \text{INT}_{n-1}(z)$  durch  $\forall w: (w = y \vee w = z) \Rightarrow \text{INT}_{n-1}(w)$  zu ersetzen.  $\square$

### 3.4.2 $(\mathcal{I}, \text{INT}) \notin \text{PAR}_{\mathbb{R}}$

Folgender in [Cuc93] bewiesene Satz charakterisiert die Beziehung zwischen der Anzahl der Zusammenhangskomponenten einer Menge  $L_n \subseteq \mathbb{R}^n$  und der mindestens notwendigen Tiefe eines algebraischen Schaltkreises  $C_n$ , der diese Menge entscheidet. Eine Zusammenhangskomponente  $K \subseteq L_n$  ist dabei eine inklusionsmaximale Teilmenge von  $L_n$  derart, dass je zwei Punkte aus  $K$  durch eine ebenfalls in der Komponente liegende stetige Kurve miteinander verbunden sind.

### 3 Reelle interaktive Beweissysteme

**Satz 3.4.3.** Sei  $(Y, L)$  ein Entscheidungsproblem mit  $L \subseteq Y \subseteq \mathbb{R}^\infty$  derart, dass  $L$  von einer  $\mathbb{P}_{\mathbb{R}}$ -uniformen Familie algebraischer Schaltkreise  $\{C_i\}_{i \in \mathbb{N}}$  über  $Y$  entscheidbar ist. Weiter sei  $L_n = L \cap \mathbb{R}^n$  und  $d(C_n)$  die Tiefe des Schaltkreises der  $L_n$  entscheidet. Dann gilt

$$d(C_n) \in \Omega \left( \sqrt{\frac{\log_2(\text{Anzahl der Zusammenhangskomponenten von } L_n)}{n}} \right). \quad \square$$

Aus diesem Satz folgt, dass  $(\mathcal{I}, \text{INT})$  nicht durch eine Familie algebraischer Schaltkreise mit polynomieller Tiefe entscheidbar ist:

**Korollar 3.4.4.** Es gilt  $(\mathcal{I}, \text{INT}) \notin \text{PAR}_{\mathbb{R}}$ .

*Beweis.* Sei  $\{C_i\}_{i \in \mathbb{N}}$  eine  $\mathbb{P}_{\mathbb{R}}$ -uniforme Familie algebraischer Schaltkreise die INT über  $\mathcal{I}$  entscheidet. Die Menge  $L_n = \{(x_1, \dots, x_{n+1}) \in \mathbb{R}^{n+1} \mid x_1 \in \mathbb{Z} \cap [0, 2^{2^n}]\}$  hat genau  $2^{2^n} + 1$  Zusammenhangskomponenten. Wegen  $\text{INT}_n \subseteq L_n$  ist dies ebenfalls eine untere Schranke für  $\text{INT}_n$  und gemäß Satz 3.4.3 folgt für die Tiefe des zugehörigen Schaltkreises  $d(C_{n+1}) \in \Omega(2^{2^n} / \sqrt{n})$ .  $\square$

*Bemerkung 3.4.5.* Gemäß Lemma 3.4.2 folgt, dass auch  $(\text{TRAO}, \text{TRAO}_+)$  nicht in  $\text{PAR}_{\mathbb{R}}$  enthalten ist. Entsprechend ist Quantorenelimination über den reellen Zahlen mit Addition und Ordnung nicht innerhalb der Komplexitätsklasse  $\text{PAR}_{\mathbb{R}}$  möglich.

#### 3.4.3 $(\mathcal{P}, \text{PROD}) \in \text{IP}_{\mathbb{R}_+}$

Folgendes Lemma belegt die Behauptung  $(\mathcal{P}, \text{PROD}) \in \text{IP}_{\mathbb{R}_+}$ .

**Lemma 3.4.6.** Der in Algorithmus 3.4 angegebene Verifizierer  $V$  ist ein  $\text{IP}_{\mathbb{R}_+}$ -Verifizierer für das Entscheidungsproblem  $(\mathcal{P}, \text{PROD})$ .

*Beweis.* Es ist für den Fall  $(x, y, 1, \dots, 1) \in \text{PROD}$  die Vollständigkeit und für den Fall  $(x, y, 1, \dots, 1) \notin \text{PROD}$  die Korrektheit des Verifizierers  $V$ , sowie seine BSS-Berechenbarkeit im additiven Modell zu beweisen.

Der Verifizierer  $V$  reduziert in insgesamt  $n$  Runden jeweils das Entscheidungsproblem  $\text{PROD}_k(x_k, y_k)$  auf  $\text{PROD}_{k-1}(x_k + rz, z + ry_k)$ , wobei  $z \in \mathbb{R}$  eine Nachricht des Beweisers und  $r \in \{1, \dots, 2n\}$  eine anschließend vom Verifizierer gewählte Zufallszahl ist. Nach Abschluss der letzten Runde kann  $V$  die Entscheidung  $\text{PROD}_0(x_0, y_0) = (x_0 = y_0 \cdot 2^{2^0}) = (x_0 = y_0 + y_0)$  schließlich selbst treffen.

(i) sei  $(x, y, 1, \dots, 1) \in \text{PROD}$

Gemäß Lemma 3.4.2 gilt die folgende Implikation

$$\text{PROD}_k(x, y) \Rightarrow \exists z: \text{PROD}_{k-1}(x, z) \wedge \text{PROD}_{k-1}(z, y).$$

---

**Algorithmus 3.4** :  $\text{IP}_{\mathbb{R}_+}$ -Verifizierer  $V$  für  $(\mathcal{P}, \text{PROD})$ 


---

**Eingabe** : Eingabevektor  $(x, y, 1, \dots, 1) \in \mathbb{R}^{n+2}$ 
**Ausgabe** : akzeptiere, falls  $x = y \cdot 2^{2^n}$ , sonst verwerfe

 setze  $x_n := x$  und  $y_n := y$ 
**for**  $k := n$  **downto** 1 **do**

 empfangen  $z \in \mathbb{R}$  vom Beweiser

 wähle Zufallszahl  $r \in \{1, \dots, 2n\}$  und sende  $r$  an den Beweiser

 setze  $x_{k-1} := x_k + r \cdot z$  und  $y_{k-1} := z + r \cdot y_k$ 

 prüfe, ob  $x_0 = y_0$ , sonst verwerfe

 akzeptiere

---

In jeder der  $n$  Runden existiert also ein  $z$ , nämlich  $z = y \cdot 2^{2^{k-1}}$ , das einen Beweis für  $\text{PROD}_k(x, y)$  auf einen Beweis für  $\text{PROD}_{k-1}(x, z)$  reduziert. Sei also für eine Runde  $k$  die Induktionsvoraussetzung  $x_k = y_k \cdot 2^{2^k} \wedge z = y_k \cdot 2^{2^{k-1}}$  erfüllt. Dann gilt  $x_k = z \cdot 2^{2^{k-1}}$  und nach  $r$ -facher Addition der gemäß Induktionsvoraussetzung gültigen Gleichung  $z = y_k \cdot 2^{2^{k-1}}$  ergibt sich folgender Induktionsschritt

$$x_k = y_k \cdot 2^{2^k} \Rightarrow \exists z \forall r: (x_k + rz) = (z + ry_k) \cdot 2^{2^{k-1}}.$$

Da nach Voraussetzung  $x_n = y_n \cdot 2^{2^n}$  gilt, folgt mit  $x_{k-1} = x_k + rz$  und  $y_{k-1} = z + ry_k$  für das in Runde  $k$  empfangene  $z$  und gewählte  $r$ , dass auch  $x_0 = y_0$  gilt. Somit existiert ein Beweiser  $P$  derart, dass  $\Pr[\langle P, V \rangle(x, y, 1, \dots, 1) = 1] = 1$ .

(ii) sei  $(x, 1, \dots, 1) \notin \text{PROD}$

Es genügt zu zeigen, dass in jeder Runde  $k$  unter der Induktionsvoraussetzung  $x_k \neq y_k \cdot 2^{2^k}$  mit hoher Wahrscheinlichkeit auch die Ungleichung  $x_{k-1} \neq y_{k-1} \cdot 2^{2^{k-1}}$ , d. h.  $x_k + rz \neq z + ry_k \cdot 2^{2^{k-1}}$  erfüllt ist. In Runde  $k$  soll also gelten

$$x_k \neq y_k \cdot 2^{2^k} \Rightarrow \forall z: \Pr_{r \in \{1, \dots, 2n\}} [x_k + rz = z + ry_k \cdot 2^{2^{k-1}}] \leq \frac{1}{2n}.$$

Durch Äquivalenzumformungen der Zufallsvariable ergibt sich die Gleichung

$$r \cdot (z - y_k \cdot 2^{2^{k-1}}) + (x_k - z \cdot 2^{2^{k-1}}) = 0.$$

Diese ist offensichtlich dann erfüllt, wenn  $z = y_k \cdot 2^{2^{k-1}} \wedge x_k = z \cdot 2^{2^{k-1}}$  gilt. Da dies nach Voraussetzung ausgeschlossen ist, gibt es für beliebige  $x_k, y_k, z \in \mathbb{R}$  jeweils höchstens ein  $r \in \mathbb{R}$  derart, dass die homogene und in der Variable  $r$  lineare Gleichung erfüllt ist. Da für dieses  $r$  nicht notwendigerweise  $r \in \{1, \dots, 2n\}$  gilt, ergibt sich die obige Abschätzung der Wahrscheinlichkeit.

### 3 Reelle interaktive Beweissysteme

In jeder der  $n$  Runden besteht also mit Wahrscheinlichkeit  $\leq \frac{1}{2^n}$  die Möglichkeit, dass trotz  $\neg \text{PROD}_k(x_k, y_k)$  ein  $x_{k-1}$  sowie ein  $y_{k-1}$  derart bestimmt werden, dass  $\text{PROD}_{k-1}(x_{k-1}, y_{k-1})$  gilt. Mit Hilfe der booleschen Ungleichung ergibt sich daher eine Wahrscheinlichkeit  $\leq \frac{1}{2} - \varepsilon$ , dass der abschließende Test  $x_0 = y_0 + y_0$  fälschlicherweise  $\text{PROD}_n(x_n, y_n)$  bestätigt.

(iii) *additive BSS-Berechenbarkeit*

Bis auf die beiden Multiplikationen  $r \cdot z$  und  $r \cdot y_k$  sind alle Berechnungsschritte des Verifizierers  $V$  offensichtlich im additiven BSS-Modell berechenbar. Da die Zufallszahl  $r$  aus der Menge  $\{1, \dots, 2n\}$  gewählt wird kann die Multiplikation einer beliebigen reellen Zahl mit  $r$  durch maximal  $2n$  Additionen simuliert werden. Insgesamt ergibt sich eine Laufzeit aus  $O(n^2)$ .  $\square$

#### 3.4.4 $(\mathcal{I}, \text{INT}) \in \text{IP}_{\mathbb{R}_+}$

Folgendes Lemma belegt die Behauptung  $(\mathcal{I}, \text{INT}) \in \text{IP}_{\mathbb{R}_+}$ .

**Lemma 3.4.7.** *Der in Algorithmus 3.5 angegebene Verifizierer  $V$  ist ein  $\text{IP}_{\mathbb{R}_+}$ -Verifizierer für das Entscheidungsproblem  $(\mathcal{I}, \text{INT})$ .*

*Beweis.* Analog zum vorhergehenden Lemma ist für den Fall  $(x, 1, \dots, 1) \in \text{INT}$  die Vollständigkeit und für den Fall  $(x, 1, \dots, 1) \notin \text{INT}$  die Korrektheit des Verifizierers  $V$ , sowie seine BSS-Berechenbarkeit im additiven Modell zu beweisen.

Der Verifizierer  $V$  empfängt zunächst vom Beweiser eine reelle Zahl  $N$  und überprüft mit Hilfe des  $\text{IP}_{\mathbb{R}_+}$ -Protokolls für  $\text{PROD}$ , ob  $N = 2^{2^n}$ , d. h. ob  $\text{PROD}(N, 1)$  gilt. Sei im Folgenden also o. B. d. A.  $1 \leq x \leq N$ . Damit genügt es zu zeigen, dass der Verifizierer für solche  $x \in \mathbb{R}$  entscheiden kann, ob  $x \in \mathbb{Z}$  gilt.

Dies gelingt dem Verifizierer, indem er in jeder der  $n$  Runden das Entscheidungsproblem  $x_k \in \mathbb{Z}$  für ein  $x_k \leq q \cdot 2^{2^k}$  auf die Frage  $x_{k-1} \in \mathbb{Z}$  für ein  $x_{k-1} \leq q \cdot 2^{2^{k-1}}$  reduziert. Dabei ist  $q$  eine geeignet in Abhängigkeit von  $n$  gewählte Konstante. Im Anschluss an die letzte Runde kann der Verifizierer das Entscheidungsproblem  $x_0 \in \mathbb{Z}$  für  $x_0 \leq q \cdot 2^{2^0} = 2q$  schließlich selbst lösen. Dazu prüft er mit Hilfe von Algorithmus 3.3, ob die Gleichung  $x_0 - [x_0] = 0$  erfüllt ist.

In Runde  $k$  ergibt sich  $x_{k-1}$  aus  $x_k$ , indem  $x_k$  zunächst (ganzzahlig) durch  $q$  dividiert wird. Das Ergebnis  $x' = x_k \text{ div } q$  wird (ebenfalls ganzzahlig) durch  $2^{2^{k-1}}$  dividiert. Der Quotient  $u$  und der Rest  $v$  dieser Division werden mit einer Zufallszahl  $r$  zu  $x_{k-1} = u + r \cdot v$  zusammengefasst. Beide Divisionen werden berechnet, indem der Beweiser Quotient und Rest zur Verfügung stellt. Der Verifizierer prüft in einer Kontrollrechnung, ob Quotient und Rest zu Dividend und Divisor passen.

---

**Algorithmus 3.5** :  $\text{IP}_{\mathbb{R}_+}$ -Verifizierer  $V$  für  $(\mathcal{I}, \text{INT})$ 


---

**Eingabe** : Eingabevektor  $(x, 1, \dots, 1) \in \mathbb{R}^{n+1}$

**Ausgabe** : akzeptiere, falls  $x \in \mathbb{Z} \cap [0, 2^{2^n}]$ , sonst verwerfe

empfangen  $N \in \mathbb{R}$  vom Beweiser

prüfe, ob  $N = 2^{2^n}$  mittels  $\text{PROD}_n(N, 1)$ , sonst verwerfe

prüfe, ob  $0 \leq x \leq N$ , sonst verwerfe

akzeptiere, falls  $x = N$

berechne  $q := (4n)!$

setze  $x_n := x$

**for**  $k := n$  **downto** 1 **do**

empfangen  $x', z \in \mathbb{R}$  vom Beweiser

prüfe, ob  $x' \geq 0$ ,  $0 \leq z < q$ ,  $z \in \mathbb{Z}$  und  $x_k = q \cdot x' + z$ , sonst verwerfe

empfangen  $u, v \in \mathbb{R}$  vom Beweiser

prüfe, ob  $u \geq 0$ ,  $v \geq 0$  und ob  $x' = 2^{2^{k-1}} \cdot u + v$  mittels  $\text{PROD}_{k-1}(x' - v, u)$ ,

sonst verwerfe

wähle Zufallszahl  $r \in \{1, \dots, 4n\}$  und sende  $r$  an den Beweiser

setze  $x_{k-1} := u + r \cdot v$

prüfe, ob  $x_0 < 2q$  und  $x_0 \in \mathbb{Z}$ , sonst verwerfe

akzeptiere

---

### 3 Reelle interaktive Beweissysteme

(i) sei  $x \in \mathbb{Z}$ , d. h.  $(x, 1, \dots, 1) \in \text{INT}$

Sei nun in Runde  $k$  per Induktionsvoraussetzung  $x_k \in \mathbb{Z}$  und  $x_k < q \cdot 2^{2^k}$ . Wegen  $x_k \in \mathbb{Z}$  kann der Beweiser ganzzahlige  $x', z \in \mathbb{Z}$  senden derart, dass  $x_k = q \cdot x' + z$  gilt. Analog gibt es auch ganzzahlige  $u, v \in \mathbb{Z}$  derart, dass  $x' = 2^{2^{k-1}} \cdot u + v$  gilt. Entsprechend ergibt sich für  $r \in \{1, \dots, 4n\}$  folgende Implikationskette

$$\begin{aligned} x_k < q \cdot 2^{2^k} &\Rightarrow x' \leq \frac{x_k}{q} < 2^{2^k} \\ &\Rightarrow u \leq \frac{x'}{2^{2^{k-1}}} < 2^{2^{k-1}} \wedge v < 2^{2^{k-1}} \\ &\Rightarrow x_{k-1} = u + r \cdot v \leq (r+1) \cdot 2^{2^{k-1}} \Rightarrow x_{k-1} < q \cdot 2^{2^{k-1}}. \end{aligned}$$

Wegen  $u, v, r \in \mathbb{Z}$  ist auch  $x_{k-1} \in \mathbb{Z}$ . Da nach Voraussetzung  $x_n \in \mathbb{Z}$  und  $x_n < q \cdot 2^{2^n}$  gilt, gibt es somit einen Beweiser  $P$ , so dass auch  $x_0 \in \mathbb{Z}$  und  $x_0 < q \cdot 2^{2^0} = 2q$  erfüllt ist. Mit diesem Beweiser ergibt sich  $\Pr[\langle P, V \rangle(x, 1, \dots, 1) = 1] = 1$ .

(ii) sei  $x \notin \mathbb{Z}$ , d. h.  $(x, 1, \dots, 1) \notin \text{INT}$

Es genügt zu zeigen, dass in jeder Runde  $k$  unter der Induktionsvoraussetzung  $x_k \notin \mathbb{Z}$  mit hoher Wahrscheinlichkeit auch  $x_{k-1} \notin \mathbb{Z}$  gilt, d. h.

$$x_k \notin \mathbb{Z} \Rightarrow \forall x', z, u, v \in \mathbb{R}: \Pr_{r \in \{1, \dots, 4n\}} [x_{k-1} \in \mathbb{Z}] \leq \frac{1}{2n}.$$

Der Verifizierer wird nur dann akzeptieren, wenn der Beweiser ein ganzzahliges  $z \in \mathbb{Z}$  sendet. Wegen  $q \in \mathbb{Z}$  muss er außerdem ein  $x' \notin \mathbb{Z}/q = \{y \mid q \cdot y \in \mathbb{Z}\}$  senden, damit  $x_k = q \cdot x' + z$  für  $x_k \notin \mathbb{Z}$  erfüllt werden kann. Des Weiteren müssen  $u, v \in \mathbb{R}$  die Bedingung  $x' = 2^{2^{k-1}} \cdot u + v$  erfüllen. Es genügt zu zeigen, dass unter diesen Voraussetzungen höchstens ein  $r$  existiert, welches  $x_{k-1} = u + r \cdot v \in \mathbb{Z}$  erfüllt.

Angenommen es gäbe zwei  $r, r' \in \{1, \dots, 4n\}$  mit  $r < r'$  derart, dass  $u + r \cdot v \in \mathbb{Z}$  und  $u + r' \cdot v \in \mathbb{Z}$  erfüllt sind. Die Differenz ist entsprechend ebenfalls ganzzahlig und es folgt  $(r' - r) \cdot v \in \mathbb{Z}$ . Da  $(r' - r)$  ein Teiler von  $q$  ist folgt weiter  $q \cdot v \in \mathbb{Z}$  und somit  $v \in \mathbb{Z}/q$ . Da nach Annahme  $u + r \cdot v \in \mathbb{Z}$  gilt folgt auch  $u \in \mathbb{Z}/q$ . Somit würde sich wegen  $x' = 2^{2^{k-1}} \cdot u + v$  im Widerspruch zu den Voraussetzungen  $x \in \mathbb{Z}/q$  ergeben.

Entsprechend ist höchstens eine der möglichen  $x_{k-1} \in \{u + r \cdot v \mid 1 \leq r \leq 4n\}$  ganzzahlig. Durch  $O(\log n)$  viele Wiederholungen kann die Fehlerwahrscheinlichkeit des beteiligten PROD-Protokolls auf  $\leq \frac{1}{4n}$  gesenkt werden. Insgesamt ergibt sich eine Wahrscheinlichkeit  $\leq \frac{1}{2n}$ , dass in einer der  $n$  Runden ein  $x_k \notin \mathbb{Z}$  fälschlicherweise auf ein  $x_{k-1} \in \mathbb{Z}$  reduziert wird. Mit der booleschen Ungleichung lässt sich die gesamte Fehlerwahrscheinlichkeit auf  $\leq \frac{1}{2} - \varepsilon$  abschätzen.

(iii) additive BSS-Berechenbarkeit

Durch die Wahl von  $q = (4n)!$  ergibt sich  $\text{bitsize}(q) \in O(n \log n)$ . Die Kontrollrechnung für die Division durch  $q$  kann daher vom Verifizierer selbst durchgeführt werden. Um die Division durch  $2^{2^{k-1}}$  zu überprüfen, wird das  $\text{IP}_{\mathbb{R}_+}$ -Protokoll für

PROD verwendet. Die Multiplikation einer reellen Zahl mit  $r \in \{1, \dots, 4n\}$  kann vom Verifizierer durch fortgesetztes Addieren simuliert werden. Da  $x_0$  von der Größenordnung  $2q$  ist, kann auch der abschließende Test  $x_0 - [x_0] = 0$  vom Verifizierer selbst berechnet werden.  $\square$

Wegen  $(\mathcal{I}, \text{INT}) \notin \text{PAR}_{\mathbb{R}}$  und  $\text{PAR}_{\mathbb{R}_+} = \text{BIP}_{\mathbb{R}_+}$  sowie  $\text{PAR}_{\mathbb{R}} \subseteq \text{BIP}_{\mathbb{R}}$  folgt somit:

**Satz 3.4.8.** *Es gilt  $\text{PAR}_{\mathbb{R}} \subseteq \text{IP}_{\mathbb{R}_+}$  sowie  $\text{BIP}_{\mathbb{R}_+} \subseteq \text{IP}_{\mathbb{R}_+}$  und  $\text{BIP}_{\mathbb{R}} \subseteq \text{IP}_{\mathbb{R}}$ .*  $\square$

### 3.4.5 $\text{IP}_{\mathbb{R}_+}$ -Entscheidbarkeit diskreter Entscheidungsprobleme

Ein diskretes Entscheidungsproblem ist o. B. d. A. eine (stets abzählbare) Teilmenge von  $\{0, 1\}^*$ , also der Menge aller endlichen Worte über  $\{0, 1\}$ . Jedes Wort  $w \in \{0, 1\}^n$  kann eineindeutig als eine natürliche Zahl  $N$  mit  $2^n \leq N < 2^{n+1}$  kodiert werden, indem  $N$  gleich dem Wert  $\text{val}(1w)$  der Binärdarstellung „ $1w$ “ gewählt wird. Sei also allgemein die Nummer eines Wortes  $w \in \{0, 1\}^*$  definiert durch  $N(w) = \text{val}(1w)$ .

Entsprechend kann jedes diskrete Entscheidungsproblem  $L$  mit  $\emptyset \subsetneq L \subseteq \{0, 1\}^*$  als reelle Zahl  $\alpha_L = \sum_{w \in L} 2^{-N(w)}$  aus dem Intervall  $(0, 1)$  dargestellt werden. In der Binärdarstellung von  $\alpha_L$  ist das  $N$ -te Bit genau dann gesetzt, wenn das  $N$ -te Wort, also das eindeutige Wort  $w$  mit  $N(w) = N$ , in  $L$  enthalten ist.

Eine BSS-Maschine mit der zu einem diskreten Entscheidungsproblem  $L$  gehörenden Maschinenkonstante  $\alpha_L$  kann  $L$  entscheiden, indem sie für eine Eingabe  $w$  prüft, ob das  $N(w)$ -te Bit in der Binärdarstellung von  $\alpha_L$  gesetzt ist. In Algorithmus 3.6 wird dazu das  $N(w)$ -te Bit durch Multiplikation mit  $2^{N(w)-1}$  auf die Position der ersten Nachkommastelle verschoben. Nach Subtraktion des ganzzahligen Anteils, kann durch Verdoppeln geprüft werden, ob das Bit gesetzt ist.

Somit ist es möglich, beliebige diskrete Entscheidungsprobleme mit dem angegebenen BSS-Algorithmus zu entscheiden. Dennoch bleibt dies ein schwieriges Problem, da eine multiplikative Maschine zwar effizient Zahlen der Größenordnung  $2^{2^n}$  erzeugen kann, jedoch die Bestimmung des ganzzahligen Anteils derselben schwierig bleibt. Unter Anwendung der im vorigen Abschnitt erzielten Resultate wird im Folgenden bewiesen, dass diese Probleme mit interaktiven Beweissystemen effizient gelöst werden können.

**Lemma 3.4.9.** *Es existiert ein  $\text{IP}_{\mathbb{R}_+}$ -Verifizierer, der für gegebene  $x, y \in \mathbb{R}$  und eine binär kodierte ganze Zahl  $N \geq 0$  entscheidet, ob  $x = y \cdot 2^N$ .*

*Beweis.* Sei  $n = \text{bitsize}(N)$  die Länge der binären Kodierung von  $N$ . Entsprechend hat  $N$  die binäre Darstellung  $N = 2^{n_k} + \dots + 2^{n_1}$  für geeignet gewählte Exponenten  $0 \leq n_1 < \dots < n_k < n$ . Somit folgt

$$x = y \cdot 2^N \Leftrightarrow x = y \cdot 2^{2^{n_k} + \dots + 2^{n_1}} \Leftrightarrow x = y \cdot 2^{2^{n_k}} \cdot \dots \cdot 2^{2^{n_1}}$$

---

**Algorithmus 3.6** : BSS-Algorithmus für diskrete Entscheidungsprobleme  $L \subseteq \{0, 1\}^*$

---

**Eingabe** : Wort  $w = (w_1, \dots, w_n) \in \{0, 1\}^*$

**Ausgabe** : akzeptiere, falls  $w \in L$ , sonst verwerfe

**Daten** : Maschinenkonstante  $\alpha_L$

berechne  $N := N(w)$  (i)

berechne  $b := \alpha_L \cdot 2^{N-1}$  (ii)

berechne  $c := b - [b]$  (iii)

prüfe, ob  $2c \geq 1$ , sonst verwerfe  
akzeptiere

---

Die letzte Bedingung kann durch insgesamt  $k$  Instanzen des PROD-Entscheidungsproblems ausgedrückt werden und es folgt

$$x = y \cdot 2^N \Leftrightarrow \exists z_{k-1}, \dots, z_1 : \text{PROD}_{n_k}(z_{k-1}, y) \wedge \dots \wedge \text{PROD}_{n_1}(x, z_1).$$

Ein Verifizierer  $V$ , der  $x = y \cdot 2^N$  überprüfen soll, kann dazu in jeder der  $k$  Runden vom Beweiser jeweils eine Zahl  $z_{i-1}$  empfangen und mit Hilfe des Verifizierers aus Algorithmus 3.4 die Bedingung  $\text{PROD}_{n_i}(z_{i-1}, z_i)$  überprüfen.  $\square$

**Korollar 3.4.10.** *Algorithmus 3.6 definiert einen  $\text{IP}_{\mathbb{R}_+}$ -Verifizierer für beliebige diskrete Entscheidungsprobleme.*

*Beweis.* Gemäß den vorigen Ausführungen genügt es zu zeigen, dass die drei Berechnungsschritte in Algorithmus 3.6 von einem  $\text{IP}_{\mathbb{R}_+}$ -Verifizierer ausgeführt werden können.

- (i) Die Berechnung der Nummer  $N = N(w) = \text{val}(1w)$  des in Binärdarstellung gegebenen Wortes  $w \in \{0, 1\}^*$  kann entfallen, da das  $\text{IP}_{\mathbb{R}_+}$ -Protokoll für Berechnungsschritt (ii) die Nummer  $N$  wiederum in ihrer Binärdarstellung benötigt.
- (ii) Das Produkt  $b$  kann berechnet werden, indem der Verifizierer vom Beweiser ein  $b \in \mathbb{R}$  empfängt und mit Hilfe des gemäß Lemma 3.4.9 existierenden  $\text{IP}_{\mathbb{R}_+}$ -Protokoll überprüft, ob  $b = \alpha_L \cdot 2^{N-1}$  erfüllt ist.
- (iii) Um den Nachkommaanteil von  $b$  zu berechnen, empfängt der Verifizierer vom Beweiser ein  $c \in \mathbb{R}$  und prüft, ob  $0 < c < 1$  erfüllt ist. Anschließend überprüft er mit dem  $\text{IP}_{\mathbb{R}_+}$ -Protokoll nach Lemma 3.4.7 die Bedingung  $\text{INT}_{n+1}(b - c)$ . Dies genügt, da  $c = b - [b]$  äquivalent zu  $b - c \in \mathbb{Z}$  ist und  $b - c < 2^N < 2^{2^{n+1}}$  gilt.  $\square$

Aus dem vorigen Korollar folgt:

**Korollar 3.4.11.** *Alle diskreten Entscheidungsprobleme  $L \subseteq \{0, 1\}^*$  sind in  $\text{IP}_{\mathbb{R}_+}$  enthalten.*  $\square$

## Zusammenfassung

In dieser Arbeit wurden die Beziehungen verschiedener, durch interaktive Beweissysteme definierter Komplexitätsklassen untersucht. Dazu wurde zunächst eine von konkreten Berechnungsmodellen abstrahierende Definition eines interaktiven Beweissystems und dessen effizienter Berechenbarkeit entwickelt.

Darauf basierend konnten im Rahmen einer einheitlichen Nomenklatur eine Reihe von Komplexitätsklassen für das klassische sowie das BSS-Berechnungsmodell definiert werden. Dabei waren das verwendete Berechnungsmodell sowie die Art der ausgetauschten Nachrichten die beiden zueinander orthogonalen Klassifikationskriterien. Im Einzelnen ergaben sich die folgenden Komplexitätsklassen:

**Im klassischen Modell** sind nur diskrete Nachrichten zwischen Verifizierer und Beweiser möglich, es ergab sich die Komplexitätsklasse  $IP$ .

**Im BSS-Modell** kann sinnvoll eingeschränkt werden, dass der Verifizierer stets diskrete Nachrichten sendet. Falls dies auch für den Beweiser gilt ergab sich die Klasse  $BIP_{\mathbb{R}}$ , sendet der Beweiser reelle Nachrichten die Klasse  $IP_{\mathbb{R}}$ .

**Im additiven BSS-Modell** ergaben sich mit der Beschränkung auf additive BSS-Berechenbarkeit analog die Komplexitätsklassen  $BIP_{\mathbb{R}_+}$  und  $IP_{\mathbb{R}_+}$ .

Aufbauend auf diesen Definitionen ergab sich die Möglichkeit die strukturellen Beziehungen der interaktiven Komplexitätsklassen untereinander und zu verschiedenen deterministischen Klassen zu untersuchen.

Die Beziehungen zu deterministischen Klassen betreffend, wurde zunächst ein bis auf die Berücksichtigung der unterschiedlichen Berechnungsmodelle einheitlicher Beweis für die Inklusion der interaktiven Klassen  $IP$  und  $BIP_{\mathbb{R}}$  in den deterministischen Klassen  $PSPACE$  und  $PAR_{\mathbb{R}}$  angegeben.

Das Ziel war anschließend, die vollständige Charakterisierung der interaktiven Komplexitätsklassen durch diese deterministischen Klassen nachzuweisen. Dazu wurde in beiden Berechnungsmodellen die einheitliche Vorgehensweise gewählt, für ein vollständiges Problem der deterministischen Klasse ein interaktives Beweissystem anzugeben und anschließend dessen Vollständigkeit und Korrektheit zu beweisen. In beiden Modellen wurde zuvor die Vollständigkeit des gewählten Problems in der jeweiligen deterministischen Komplexitätsklasse nachgewiesen.

Aus didaktischen Gründen wurde eine zweigeteilte Darstellung des im klassischen Berechnungsmodell seit den 1990er Jahren bekannten Ergebnisses  $PSPACE \subseteq IP$

## Zusammenfassung

von Shamir gewählt. Der Beweis für dieses Ergebnis wurde als Erweiterung eines Beweises für die Inklusion  $\text{co-NP} \subseteq \text{IP}$  präsentiert. Insgesamt ergab sich so, als Folgerung des Resultats von Shamir, die vollständige Charakterisierung der klassischen interaktiven Beweissysteme durch die deterministische Komplexitätsklasse PSPACE.

Ein analoges Ergebnis im reellen Fall wurde für das additive BSS-Modell vorgestellt. In umfangreichen und ausführlichen Darstellungen wurde der seinerseits auf dem Resultat von Shamir beruhende Beweis von Ivanov und de Rougemont für die Inklusion  $\text{BIP}_{\mathbb{R}_+} \subseteq \text{PAR}_{\mathbb{R}_+}$  wiedergegeben. Dazu wurden in einem separaten Abschnitt zuvor die benötigten mathematischen Hilfsmittel vorgestellt, darunter das Ergebnis von Tarski über die effektive Quantorenelimination im geordneten reellen Zahlkörper. Insgesamt ergab sich so erneut die vollständige Charakterisierung einer interaktiven durch eine deterministische Komplexitätsklasse, speziell  $\text{BIP}_{\mathbb{R}_+} = \text{PAR}_{\mathbb{R}_+}$ .

Als ein Ergebnis zur Beziehung der reellen interaktiven Komplexitätsklassen untereinander wurde ein weiteres Resultat von Ivanov und de Rougemont vorgestellt. Es belegt die Separation der resultierenden Klassen, wenn der Beweiser nicht diskrete sondern reelle Nachrichten senden darf. Dazu wurde für ein geeignet gewähltes Problem zunächst ein  $\text{IP}_{\mathbb{R}_+}$ -Verifizierer angegeben und anschließend, unter Verweis auf ein Ergebnis von Cucker, dessen trennender Charakter bezüglich der Klasse  $\text{BIP}_{\mathbb{R}}$  nachgewiesen.

Eine Folgerung dieses Separationsresultats war, dass Quantorenelimination über den reellen Zahlen mit Addition und Ordnung die Mächtigkeit der Komplexitätsklasse  $\text{BIP}_{\mathbb{R}}$  übersteigt, jedoch zumindest in Einzelfällen unter  $\text{IP}_{\mathbb{R}_+}$ -Beschränkungen möglich ist.

In Vorbereitung auf die strukturellen Untersuchungen insbesondere der reellen interaktiven Komplexitätsklassen, wurde am Anfang dieser Arbeit eine Einführung in die reelle Komplexitätstheorie basierend auf dem Berechnungsmodell von Blum, Shub und Smale gegeben. Zur Fundierung der durch das BSS-Modell konstituierten Komplexitätstheorie wurde für die reelle Klasse  $\text{NP}_{\mathbb{R}}$  ein ausführlicher Nachweis der Existenz vollständiger Entscheidungsprobleme geführt.

### Weitere interessante Fragestellungen

Die in dieser Arbeit vorgestellten Ergebnisse zeichnen ein grundlegendes Bild der strukturellen Beziehungen reeller interaktiver Beweissysteme und deterministischer Komplexitätsklassen. Eine interessante Fragestellung wäre, ob sich das bisher erzielte Äquivalenzresultat  $\text{BIP}_{\mathbb{R}_+} = \text{PAR}_{\mathbb{R}_+}$  auf mächtigere Berechnungsmodelle übertragen lässt. Ein erster Schritt wäre es, an Stelle des vollen zunächst das lineare BSS-Modell zu betrachten.

Im linearen BSS-Modell ist es zusätzlich zur Addition und Subtraktion möglich mit einer endlichen Anzahl von reellen Maschinenkonstanten skalar zu multiplizieren. Entsprechend haben alle Zwischenresultate einer BSS-Maschine mit der Eingabe  $x_1, \dots, x_n$  und den Maschinenkonstanten  $\alpha_1, \dots, \alpha_k$  für  $k$ -variate Polynome  $p_1, \dots, p_n$  und  $q$  mit ganzzahligen Koeffizienten die Form

$$\sum_{i=1}^n p_i(\alpha_1, \dots, \alpha_k) x_i + q(\alpha_1, \dots, \alpha_k)$$

Für ein äquivalentes Ergebnis im linearen Modell müsste zunächst ein  $\text{PAR}_{\mathbb{R}_{\text{lin}}}$ -vollständiges Problem identifiziert werden. Anschließend könnte der Beweis von Ivanov und de Rougemont möglicherweise zu  $\text{BIP}_{\mathbb{R}_{\text{lin}}} = \text{PAR}_{\mathbb{R}_{\text{lin}}}$  erweitert werden.

Daneben könnte – die Folgerung aus dem Separationsresultat von Ivanov und de Rougemont aufgreifend – untersucht werden, in welcher Komplexitätsklasse die Quantorenelimination im additiven Modell enthalten ist. Daraus folgt die interessante Fragestellung, ob es in dieser Klasse vollständige Probleme gibt und ob diese Klasse  $\text{IP}_{\mathbb{R}_+}$  entspricht.



## Literaturverzeichnis

- [AB08] ARORA, SANJEEV und BOAZ BARAK: *Complexity Theory: A Modern Approach (DRAFT)*. Princeton University, Department of Computer Science, 2008.
- [AKS04] AGRAWAL, MANINDRA, NEERAJ KAYAL und NITIN SAXENA: *PRIMES is in P*. *Annals of Mathematics*, 160(2):781–793, 2004.
- [Bab85] BABAI, LASZLO: *Trading group theory for randomness*. In: *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, Seiten 421–429, New York, 1985. ACM.
- [Bab90] BABAI, LASZLO: *E-mail and the Unexpected Power of Interaction*. Technischer Bericht 90-15, University of Chicago, Department of Computer Science, 1990.
- [Bab94] BABAI, LASZLO: *Transparent Proofs and Limits to Approximation*. Technischer Bericht 94-07, University of Chicago, Department of Computer Science, 1994.
- [BC94] BOVET, DANIEL PIERRE und PIERLUIGI CRESCENZI: *Introduction to the theory of complexity*. Prentice Hall International (UK) Ltd., Hertfordshire, 1994.
- [BCR87] BOCHNAK, JACEK, MICHEL COSTE und MARIE-FRANÇOISE ROY: *Géométrie algébrique réelle*, Band 12 der Reihe *Ergebnisse der Mathematik und ihrer Grenzgebiete*, 3. Folge. Springer, 1987.
- [BCSS98] BLUM, LENORE, FELIPE CUCKER, MICHAEL SHUB und STEVE SMALE: *Complexity and real computation*. Springer, New York, 1998.
- [Bor77] BORODIN, ALLAN: *On Relating Time and Space to Size and Depth*. *SIAM Journal on Computing*, 6(4):733–744, 1977.
- [Brö95] BRÖCKER, LUDWIG: *Semialgebraische Geometrie*. In: GEYER, WULF-DIETER (Herausgeber): *Jahresbericht der Deutschen Mathematiker-Vereinigung*, Band 4-97, Seiten 130–156, Stuttgart, 1995. B. G. Teubner.
- [BSS89] BLUM, LENORE, MICHAEL SHUB und STEVE SMALE: *On a Theory of Computation and Complexity over the Real Numbers: NP Completeness, Recursive Functions and Universal Machines*. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.

## Literaturverzeichnis

- [CB07] CUCKER, FELIPE und IRÉNÉE BRIQUEL: *A note on parallel and alternating time*. *Journal of Complexity*, 23(4-6):594–602, 2007. Festschrift for the 60th Birthday of Henryk Wozniakowski.
- [CK94] CUCKER, FELIPE und PASCAL KOIRAN: *Computing over the reals with addition and order: higher complexity classes*. Technischer Bericht 94-08, DIMACS, 1994.
- [CKS81] CHANDRA, ASHOK KUMAR, DEXTER CAMPBELL KOZEN und LARRY JOSEPH STOCKMEYER: *Alternation*. *J. ACM*, 28(1):114–133, 1981.
- [Coo71] COOK, STEPHEN ARTHUR: *The complexity of theorem-proving procedures*. In: *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, Seiten 151–158, New York, 1971. ACM.
- [Cuc93] CUCKER, FELIPE: *On the Complexity of Quantifier Elimination: the Structural Approach*. *The Computer Journal*, 36(5):400–408, 1993.
- [DK00] DU, DING-ZHU und KER-I KO: *Theory of computational complexity*. Wiley, New York, 2000.
- [dN06] NAUROIS, PAULIN JACOBÉ DE: *A Measure of Space for Computing over the Reals*. CoRR, abs/cs/0603017, 2006.
- [FGM<sup>+</sup>89] FÜRER, MARTIN, ODED GOLDREICH, YISHAY MANSOUR, MICHAEL SIPSER und STATHIS ZACHOS: *On completeness and soundness in interactive proof systems*. In: MICALI, SILVIO (Herausgeber): *Advances in Computing Research: A Research Annual*, Band 5 (Randomness and Computation), Seiten 429–442, 1989.
- [GMR89] GOLDWASSER, SHAFI, SILVIO MICALI und CHARLES RACKOFF: *The knowledge complexity of interactive proof systems*. *SIAM Journal on Computing*, 18(1):186–208, 1989.
- [GMS87] GOLDREICH, ODED, YISHAY MANSOUR und MICHAEL SIPSER: *Interactive proof systems: Provers that never fail and random selection (Extended Abstract)*. In: *28th Annual Symposium on Foundations of Computer Science*, Seiten 449–461. IEEE, 1987.
- [GS86] GOLDWASSER, SHAFI und MICHAEL SIPSER: *Private coins versus public coins in interactive proof systems*. In: *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing*, Seiten 59–68, New York, 1986. ACM.
- [IdR98] IVANOV, SERGEI und MICHEL DE ROUGEMONT: *Interactive Protocols on the Reals*. In: *STACS '98: Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science*, Seiten 499–510, London, 1998. Springer.
- [Kat07] KATZ, JONATHAN: *CMSC 652 – Complexity Theory*. Lecture Note 11, University of Maryland, Department of Computer Science, 2007.

- [LFKN92] LUND, CARSTEN, LANCE FORTNOW, HOWARD KARLOFF und NOAM NISAN: *Algebraic methods for interactive proof systems*. J. ACM, 39(4):859–868, 1992.
- [Mee95] MEER, KLAUS: *Eine Einführung in die reelle Komplexitätstheorie*. Vorlesungsskript, RWTH Aachen, 1995.
- [Mic89] MICHAUX, CHRISTIAN: *Une remarque à propos des machines sur  $\mathbb{R}$  introduites par Blum, Shub et Smale*. Comptes Rendus de l'Académie des sciences Paris, 309, Série I:435–437, 1989.
- [MM97] MEER, KLAUS und CHRISTIAN MICHAUX: *A survey on real structural complexity theory*. Bulletin of the Belgian Mathematical Society, 4(1):113–148, 1997.
- [MÖ04] MICHAUX, CHRISTIAN und ADEM ÖZTÜRK: *Quantifier Elimination following Muchnik*. Preprint #10, Université de Mons-Hainaut, Institut de Mathématique, 2004.
- [Pap94] PAPADIMITRIOU, CHRISTOS HARILAOS: *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.
- [Rab76] RABIN, MICHAEL OSER: *Probabilistic algorithms*. In: TRAUB, JOSEPH F. (Herausgeber): *Algorithms and Complexity, Recent Results and New Directions*, Seiten 21–39, New York, 1976. Academic Press.
- [Ren92] RENEGAR, JAMES: *On the computational complexity and geometry of the first-order theory of the reals. Parts I, II and III*. Journal of Symbolic Computation, 13(3):255–352, 1992.
- [Sav70] SAVITCH, WALTER: *Relationships between nondeterministic and deterministic tape complexities*. Journal of Computer and System Sciences, 4(2):177–192, 1970.
- [Sch86] SCHRIJVER, ALEXANDER: *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, 1986.
- [Sei54] SEIDENBERG, ABRAHAM: *A New Decision Method for Elementary Algebra*. The Annals of Mathematics, Second Series, 60(2):365–374, 1954.
- [Sha92] SHAMIR, ADI: *IP = PSPACE*. J. ACM, 39(4):869–877, 1992.
- [She92] SHEN, ALEXANDER: *IP = PSPACE: simplified proof*. J. ACM, 39(4):878–880, 1992.
- [Sip06] SIPSER, MICHAEL: *Introduction to the theory of computation*. Thomson Course Technology, Boston, 2. Auflage, 2006.
- [Son85] SONTAG, EDUARDO: *Real Addition And The Polynomial Hierarchy*. Information Processing Letters, 20:115–120, 1985.

## Literaturverzeichnis

- [SP98] SCHÖNING, UWE und RANDALL JAMES PRUIM: *Gems of Theoretical Computer Science*. Springer, Berlin, 1998.
- [SW70] STOER, JOSEF und CHRISTOPH WITZGALL: *Convexity and Optimization in Finite Dimensions I*, Band 163 der Reihe *Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen*. Springer, Berlin, 1970.
- [Tar51] TARSKI, ALFRED: *A decision method for elementary algebra and geometry*. Univ. Calif. Press, Berkeley, 2. Auflage, 1951.