NP-vollständige Probleme im Blum-Shub-Smale Rechenmodell über $\mathbb{R}, \mathbb{C}$ und $\mathbb{Z}_2$ Zur Entscheidbarkeit von NP

Ausarbeitung eines Seminarvortrags von Markus Schwagenscheidt

Sommersemester 2011

Inhaltsverzeichnis

1 Einleitung

Bei dieser Arbeit handelt es sich um die Aussarbeitung eines Vortrages im Rahmen des Seminars Reelle Komplexität, das im Sommersemester 2011 von Herr Prof. Dr. Martin Ziegler und Frau Dr. Ulrike Brand an der Technischen Universität Darmstadt angeboten wurde.

Wir wollen Entscheidungsprobleme über den reellen Zahlen $\mathbb{R}$, den komplexen Zahlen $\mathbb{C}$ und dem endlichen Körper $\mathbb{Z}_2 = \{0, 1\}$ betrachten. Über $\mathbb{R}$ kann man zum Beispiel fragen, ob ein Polynom mit reellen Koeffizienten eine reelle Nullstelle hat. Ein bekanntes Problem über $\mathbb{Z}_2$ ist das Erfüllbarkeitsproblem für aussagenlogische Formeln: Gegeben eine Formel ϕ, hat ϕ eine erfüllende Belegung?

Um Probleme über Körpern (oder allgemeiner Ringen) präzise formulieren und untersuchen zu können, verwenden wir das Blum-Shub-Smale-Rechenmodell, kurz BSS-Modell, das in [2] eingeführt wurde. Dieses Modell sowie die Komplexitätsklassen $\mathsf{P}_\mathrm{R}, \mathsf{NP}_\mathrm{R}$ und EXP_R über Ringen und den Begriff der NP_R-Vollständigkeit wollen wir im ersten Abschnitt erklären. Danach präsentieren wir im zweiten Abschnitt einige Probleme über $\mathbb{R}, \mathbb{C}$ und $\mathbb{Z}_2$ und zeigen, dass sie NP_R-vollständig sind. Im dritten Abschnitt wollen wir mithilfe des Satzes von Tarski über die Quantorenelimination für $\mathbb{R}$ und des Hilbertschen Nullstellensatzes zeigen, dass alle Probleme in $\mathsf{NP}_\mathbb{R}, \mathsf{NP}_\mathbb{C}$ und $\mathsf{NP}_{\mathbb{Z}_2}$ entscheidbar sind. Dabei halten wir uns hauptsächlich an das Buch [1] und den Artikel [3].

2 Das Blum-Shub-Smale Rechenmodell

Zur Beschreibung von Problemen und Algorithmen über einem Ring R erklären wir in diesem Abschnitt die BSS-Maschine. Dabei werden wir für R meistens die Körper $\mathbb{R}, \mathbb{C}$ oder $\mathbb{Z}_2$ wählen, wobei $\mathbb{R}$ mit der üblichen $<$ Ordnung ausgestattet ist. Andere interessante Ringe sind $\mathbb{Q}$ und $\mathbb{Z}$ sowie die endlichen Ringe $\mathbb{Z}_m = \mathbb{Z}/m\mathbb{Z}$ für $m \in \mathbb{Z}$. Die wesentliche Eigenschaft einer BSS-Maschine über einem Ring R ist, dass sie Elemente aus R speichern und vergleichen kann sowie exakt mit Ringelementen rechnen kann, also die elementaren Operationen Addition, Subtraktion und Multiplikation (und Division, falls R ein Körper ist) exakt durchführen kann. Bevor wir die genaue Definition einer BSS-Maschine geben können, benötigen wir noch ein paar Notationen und Begriffe:

2.1 Definitionen und Notationen

Es bezeichnet R^∞ die Menge aller endlichen Folgen mit Gliedern in R, das heißt, R^∞ ist die disjunkte Vereinigung aller R^n. Für jedes $x \in \mathrm{R}^\infty$ gibt es daher ein $n \in \mathbb{N}$ mit $x = (x_1, \ldots, x_n) \in \mathrm{R}^n$. Wir nennen $|x| := n$ die Länge von x.

Ein Polynom über R in n Variablen $X_1, \ldots, X_n$ ist der Form

$$f(X_1, \ldots, X_n) = \sum_{\alpha_1, \ldots, \alpha_n \in \mathbb{N}} c_{\alpha_1, \ldots, \alpha_n} X_1^{\alpha_1} \cdots X_n^{\alpha_n},$$

wobei die Koeffizienten $c_{\alpha_1, \ldots, \alpha_n} \in \mathrm{R}$ nur endlich oft von 0 verschieden sind. Um die Notation zu vereinfachen, verwenden wir die Multiindex-Schreibweise. Ein Multiindex

$\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}_0^n$ ist ein n-Tupel natürlicher Zahlen, seine Länge ist $|\alpha| = \alpha_1 + \cdots + \alpha_n$. Wir schreiben $X^\alpha := X_1^{\alpha_1} \cdots X_n^{\alpha_n}$, so dass sich f vereinfacht zu

$$f(X_1, \ldots, X_n) = \sum_{\alpha \in \mathbb{N}_0^n} c_\alpha X^\alpha.$$

Ein Summand $c_\alpha X^\alpha$ von f heißt Monom. Der Grad eines Monoms $c_\alpha X^\alpha$ ist $|\alpha|$ und der Grad von f ist das Maximum der Grade aller Monome von f (mit von 0 verschiedenen Koeffizienten). Ein Polynom vom Grad höchstens 2 nennen wir quadratisch[1]. Die Menge aller Polynome mit Koeffizienten aus R in n Variablen schreiben wir als $\mathrm{R}[X_1, \ldots, X_n]$. Ein Polynom ist zunächst eine Abbildung von R^n nach R. Wir wollen Polynome im Folgenden als Funktionen auf R^∞ auffassen, indem wir Eingaben $x \in \mathrm{R}^\infty$ für ein Polynom $f \in \mathrm{R}[X_1, \ldots, X_n]$ gegebenenfalls durch Anhängen von Nullen auf Länge n bringen oder nach Index n abschneiden. Eine rationale Funktion f ist von der Form $f = g/h$ für zwei Polynome $g, h \in \mathrm{R}[X_1, \ldots, X_n]$. Beachte, dass f an den Nullstellen von h nicht definiert ist.

Beispiel 2.1. Die Funktion

$$f(X_1, X_2, X_3) = X_1 X_2 + \sqrt{2} X_2 X_3^2 = X^{(1,1,0)} + \sqrt{2} X^{(0,1,2)}$$

ist ein Polynom aus $\mathbb{R}[X_1, X_2, X_3]$ vom Grad 3. Wollen wir f zum Beispiel an den beiden Stellen $(1, 0, 3, 7, 5), (3, 4) \in \mathbb{R}^\infty$ auswerten, so berechnen wir $f(1, 0, 3)$ und $f(3, 4, 0)$.

Es heißt $f = (f_1, f_2, \ldots) : \mathrm{R}^\infty \to \mathrm{R}^\infty$ eine polynomiale Abbildung, falls es ein $k \in \mathbb{N}$ gibt, so dass $f_1, \ldots, f_k$ Polynome in endlich vielen Variablen sind und $f_j(X) = X_j$ für alle $j > k$ gilt. Sind $f_1, \ldots, f_k$ rationale Funktionen, so heißt f eine rationale Abbildung. Eine polynomiale bzw. rationale Abbildung besteht also in endlich vielen Komponenten aus Polynomen bzw. rationalen Funktionen und lässt alle anderen Komponenten unverändert.

Beispiel 2.2. Die Abbildung

$$f(X_1, X_2, X_3, X_4, \ldots) = (X_1^2 X_2, X_2 X_3, X_3, X_4, \ldots).$$

ist eine polynomiale Abbildung mit $f_1(X) = X_1^2 X_2$, $f_2(X) = X_2 X_3$ und $f_j(X) = X_j$ für $j > 2$. Dafür schreiben wir auch kürzer $f(X_1, X_2, X_3) = (X_1^2 X_2, X_2 X_3)$.

2.2 Die BSS-Maschine

Wir können jetzt die grundlegende Definition einer BSS-Maschine über einem Ring R geben. In dieser Form findet sich die Definition in [3]:

[1] Somit nennen wir auch Polynome vom Grad 0 oder 1 quadratisch, was zwar nicht unbedingt üblich, aber für unsere Zwecke bequem ist.

Definition 2.3. Eine Maschine M über R besteht aus einem Eingaberaum $I = \mathrm{R}^\infty$, einem Ausgaberaum $O = \mathrm{R}^k, k \leq \infty$, und einem Zustandsraum $S = \mathbb{N} \times \mathbb{N} \times \mathrm{R}^\infty$ sowie einem endlichen, zusammenhängenden und gerichteten Graphen, dessen mit $1, \ldots, N$ beschriftete Knoten die Operationen der Maschine wiederspiegeln. Es gibt fünf Typen von Knoten, wobei jedem Knoten zwei Abbildungen – die Operation g des Knotens und die Nachfolgerabbildung β – zugeordnet ist:

- *Eingabeknoten:* Es gibt genau einen Eingabeknoten, und er ist mit 1 beschriftet. Zu ihm gehört die Abbildung $g_1 : I \to S$, die ein Element aus dem Eingaberaum an die Maschine übergibt. Er hat den eindeutigen Nachfolger $\beta(1)$.

- *Ausgabeknoten:* Es gibt genau einen Ausgabeknoten, und er ist mit N beschriftet. Ihm ist die Abbildung $g_N : S \to O$ zugeordnet, die das Ergebnis der Rechnung im Ausgaberaum ausgibt. Er hat keine Nachfolger. Wir setzen $\beta(N) = N$.

- *Berechnungsknoten:* Ein Berechnungsknoten m hat genau eine ausgehende Kante, er hat also einen eindeutigen Nachfolger $\beta(m)$. Zu m gehört eine Abbildung $g_m : S \to S$ der Form $g(i, j, z) = (i'(i), j'(j), z'(z))$, wobei $i'(i)$ entweder $i + 1$ oder 1 ist, $j'(j)$ entweder $j + 1$ oder 1 ist und z' eine polynomiale Abbildung ist (z' darf eine rationale Abbildung sein, falls R ein Körper ist).

- *Verzweigungsknoten:* Jeder Verzweigungsknoten m besitzt zwei mögliche Nachfolger $\beta^+(m)$ und $\beta^-(m)$ sowie die Abbilung $g_m(x) = x$. Ist R geordnet, so wird $\beta^+(m)$ gewählt, falls $x_1 \geq 0$ ist, und $\beta^-(m)$, falls $x_1 < 0$ ist. Wenn R nicht geordnet ist, so wird statt $x_1 \geq 0$ und $x_1 < 0$ auf $x_1 = 0$ und $x_1 \neq 0$ geprüft.

- *Bewegungsknoten:* Ein Knoten m dieser Art hat einen eindeutigen Nachfolger $\beta(m)$. Er ersetzt im aktuellen Zustand $(i, j, z_1, \ldots)$ das Element z_j durch z_i an der j-ten Stelle und lässt den Zustand ansonsten unverändert.

Die Rechnung einer Maschine M über R können wir anhand des Graphen von M beschreiben: Bei Eingabe $x \in I$ ist der Knoten mit der Nummer 1 aktiv und der Zustand von M ist $g_1(x)$. In jedem Rechenschritt wird der nächste Knoten m gemäß der Nachfolgerabbildung β gewählt und g_m auf den aktuellen Zustand $y \in S$ angewendet. Wir sagen, die Rechnung hält auf x, wenn die Maschine bei Eingabe x irgendwann den Knoten N erreicht. In diesem Fall ist die Ausgabe von M gleich dem Ergebnis von g_N auf dem letzten Zustand.

Die Menge aller $x \in I$, für die eine Maschine M hält, heißt Haltemenge von M und wird mit Ω_M bezeichnet. Wir können damit die Eingabe-Ausgabe-Funktion $\varphi_M : \Omega_M \to O$ von M erklären, die einer Eingabe $x \in \Omega_M$ das Ergebnis der Rechnung von M im Ausgaberaum O zuordnet.

Eine Funktion $f : \mathrm{R}^\infty \to \mathrm{R}^\infty$ heißt berechenbar, falls es eine Maschine M gibt mit $f = \varphi_M$. In diesem Fall wird also f durch M berechnet. Ferner heißt eine Menge $A \subseteq \mathrm{R}^\infty$ entscheidbar, wenn die charakteristische Funktion $\chi_A : \mathrm{R}^\infty \to \mathrm{R}$ berechenbar ist. Wir nennen A semi-entscheidbar, falls A die Haltemenge einer Maschine M ist, das heißt $A = \Omega_M$.

In [1] und in vorangegangenen Vorträgen finden sich einige Beispiele für BSS-Maschinen. Zum Beispiel kann die Nullstellensuche für eine differenzierbare Funktion $f : \mathbb{C} \to \mathbb{C}$ mittels Newton-Verfahren durch eine BSS-Maschine über $\mathbb{R}^2$ implementiert werden, die einen Startwert $z_0 = (x_0, y_0) \in \mathbb{R}^2$ und eine Fehlerschranke $\varepsilon > 0$ erhält und solange gemäß

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$$

iteriert, bis $|f(z_k)| < \varepsilon$ ist. Es kann natürlich auch passieren, dass die Fehlerschranke nie erreicht wird und die Maschine nicht hält. In einem vorangegangenen Vortrag wurden und Ω_M und φ_M näher untersucht und es wurde zum Beispiel gezeigt, dass die Menge aller Startpunkte z_0, für die das Newton-Verfahren hält, unentscheidbar ist, und dass die Exponentialfunktion nicht von einer BSS-Maschine berechenbar ist.

2.3 Komplexitätsklassen, Reduktionen und NP_R-Vollständigkeit

Ist M eine Maschine über R und $x \in \mathrm{R}^\infty$, so schreiben wir $T_M(x)$ für die Anzahl der Knoten, die während der Rechnung von M bei Eingabe x bis zum Erreichen des Ausgangsknotens besucht werden. Wir wollen $T_M(x)$ als Laufzeit von M ansehen. Jetzt können wir grundlegende Komplexitätsklassen für Entscheidungsprobleme erklären:

Definition 2.4.

- Eine Sprache $L \subseteq \mathrm{R}^\infty$ liegt in P_R, falls es ein Polynom p und eine Maschine M gibt, die L entscheidet und für die $T_M(x) \leq p(|x|)$ für alle $x \in \mathrm{R}^\infty$ gilt.
- Eine Sprache $L \subseteq \mathrm{R}^\infty$ liegt in EXP_R, falls es ein Polynom p und eine Maschine M gibt, die L entscheidet und für die $T_M(x) \leq 2^{p(|x|)}$ für alle $x \in \mathrm{R}^\infty$ gilt.
- Eine Sprache $L \subseteq \mathrm{R}^\infty$ liegt in NP_R, falls es Polynome p, q und eine Maschine M gibt mit

  $$x \in L \quad \Leftrightarrow \quad \exists w \in \mathrm{R}^{\leq q(|x|)} : \ M(x, w) = 1$$

  und $T_M(x, w) \leq p(|x|)$ für alle $x \in \mathrm{R}^\infty$.

Der Vektor w aus der obigen Definition von NP_R wird als Zeuge für x bezeichnet. Man beachte, dass alle Probleme in P_R und EXP_R per Definition entscheidbar sind. Für NP_R ist das aber nicht sofort klar und im Allgemeinen (für beliebige Ringe R) auch nicht richtig. Wir werden uns im dritten Abschnitt noch näher damit befassen.

Die nächste Definition erklärt, wann wir ein Berechnungsproblem als einfach ansehen:

Definition 2.5. Wir sagen, eine Funktion $f : \mathrm{R}^\infty \to \mathrm{R}^\infty$ ist in polynomieller Zeit berechenbar, wenn es ein Polynom p und eine Maschine M gibt, die f berechnet und die $T_M(x) \leq p(|x|)$ für alle $x \in \mathrm{R}^\infty$ erfüllt.

Um den Schwierigkeitsgrad von Problemen in NP_R vergleichen zu können, führen wir den Begriff der polynomiellen Reduktion ein:

Definition 2.6. Wir schreiben $L' \leq_p L$ für zwei Sprachen $L, L' \subseteq \mathrm{R}^\infty$, falls es eine in polynomieller Zeit berechenbare Funktion $\varphi : \mathrm{R}^\infty \to \mathrm{R}^\infty$ gibt mit

$$x \in L' \;\Leftrightarrow\; \varphi(x) \in L.$$

Eine solche Funktion φ heißt polynomielle Reduktion von L' auf L. Gilt $L' \leq_p L$ für alle $L' \in \mathsf{NP}_\mathrm{R}$, so ist L NP_R-schwer. Eine NP_R-schwere Sprache, die zusätzlich selbst in NP_R liegt, nennen wir NP_R-vollständig.

Die NP_R-vollständigen Probleme sind die schwersten Probleme in NP_R in dem Sinne, dass sich jedes Problem aus NP_R in polynomieller Zeit auf ein vollständiges Problem zurückführen lässt. Daher könnte man mit einem Polynomialzeit-Algorithmus für ein vollständiges Problem schon jedes NP_R-Problem in polynomieller Zeit lösen. Genau wie im klassichen Fall der Turing-Maschine zeigt man:

Lemma 2.7. *Es gelten die folgenden Aussagen:*

a) Die $\leq_p$-Relation ist transitiv, das heißt es gilt für alle Sprachen A, B, C:

$$A \leq_p B \wedge B \leq_p C \quad \Rightarrow \quad A \leq_p C.$$

b) Die Klassen $\mathsf{P}_\mathrm{R}, \mathsf{NP}_\mathrm{R}$ und EXP_R sind abgeschlossen unter polynomieller Reduktion. Das heißt: Ist $\mathcal{C}$ eine der drei Klassen, so folgt aus $L \in \mathcal{C}$ und $L' \leq_p L$ schon $L' \in \mathcal{C}$.

c) Ist L entscheidbar und $L' \leq_p L$, so ist auch L' entscheidbar.

d) Es gibt genau dann $\mathsf{P}_\mathrm{R} = \mathsf{NP}_\mathrm{R}$, wenn ein NP_R-vollständiges Problem in P_R liegt.

3 NP_R-vollständige Probleme über $\mathbb{R}, \mathbb{C}$ und $\mathbb{Z}_2$

Wir wollen nun einige Probleme über den reellen und komplexen Zahlen sowie über $\mathbb{Z}_2 = \{0, 1\}$ einführen und ihre NP_R-Vollständigkeit zeigen. Dabei steht R ab jetzt immer für einen der Körper $\mathbb{R}, \mathbb{C}$ oder $\mathbb{Z}_2$, wobei $\mathbb{R}$ durch $<$ geordnet ist. Eine BSS-Maschine über $\mathbb{R}$ kann also für $x = (x_1, \ldots, x_n) \in \mathbb{R}^\infty$ prüfen, ob $x_1 < 0$ oder $x_1 \geq 0$ ist und dementsprechend verzweigen. Sie kann aber auch auf Gleichheit prüfen, indem sie sowohl $x_1 < 0$ und $-x_1 < 0$ testet.

3.1 Darstellung von Polynomen

Die Probleme, mit denen wir uns beschäftigen wollen, behandeln Fragen nach der Lösbarkeit polynomieller Gleichungen und Ungleichungen. Um ein Polynom f als Eingabe an eine BSS-Maschine übergeben zu können, müssen wir f als Element von R^∞ kodieren: Nehmen wir dazu an, $f = \sum_{|\alpha| \leq d} c_\alpha X^\alpha$ sei ein Polynom vom Grad d in n Variablen. Dann schreiben wir ein Monom $c_\alpha X^\alpha$ mit $c_\alpha \neq 0$ als $d+1$-Tupel, bestehend aus c_α und den Indizes der Variablen in X^α. Zum Beispiel hat $\pi X_1^2 X_2$ die Darstellung $(\pi, 1, 1, 2)$. Falls $|\alpha| < d$ ist, so setzen wir die freien Plätze des Tupels auf 0. Die Kodierung von

f besteht nun aus d, n und der Sequenz der Kodierungen der Monome. So wird zum Beispiel das Polynom

$$5X_1^3X_2 + 7X_1^2X_2 = 5X_1X_1X_1X_2 + 7X_1X_1X_2 \quad \text{zu} \quad (4, 2, 5, 1, 1, 1, 2, 7, 1, 1, 2, 0).$$

Es gibt noch weitere sinnvolle Möglichkeiten, Polynome zu kodieren. Man kann zum Beispiel ein Polynom f vom Grad d auch darstellen, indem man für alle möglichen Monome vom Grad höchstens d die Koeffizienten angibt. Dabei werden im Allgemeinen viele Koeffizienten 0 sein. Das Polynom $X_1^3 = 0 + 0 \cdot X_1 + 0 \cdot X_1^2 + 1 \cdot X_1^3$ hat dann zum Beispiel die Darstellung $(0, 0, 0, 1)$. Im Folgenden nehmen wir die erste Kodierung.

3.2 Semi- und quasi-algebraische Formeln, 3KNF-Formeln

Über $\mathbb{R}$ betrachten wir polynomielle Gleichungen und Ungleichungen der Form $f(x) = 0$ und $f(x) < 0$, über $\mathbb{C}$ und $\mathbb{Z}_2$ stattdessen $f(x) = 0$ und $f(x) \neq 0$. Eine boolesche Kombination solcher Gleichungen und Ungleichungen nennen wir eine semi-algebraische (Ungleichungen mit $<$) bzw. quasi-algebraische (keine Ungleichungen mit $<$) Formel. Zum Beispiel ist

$$(2xy - 1 = 0 \ \vee \ x^2 < 0) \ \wedge \ \neg(4x + xy^2 = 0)$$

eine semi-algebraische Formel über $\mathbb{R}$, und

$$(x^2 + 1 \neq 0) \ \wedge \ (xy - 1 = 0)$$

eine quasi-algebraische Formel über $\mathbb{Z}_2$. Wir gehen im Folgenden davon aus, dass eine solche Formel stets in einer gewissen Normalform gegeben ist, und zwar als Konjunktion von Formeln der Art

$$f_1(x) < 0 \ \vee \cdots \vee \ f_k(x) < 0 \ \vee \ g_1(x) = 0 \ \vee \cdots \vee \ g_\ell(x) = 0,$$

mit Polynomen $f_1, \ldots, f_k, g_1, \ldots, g_\ell$, wobei über $\mathbb{C}$ und $\mathbb{Z}_2$ jedes $<$ durch ein $\neq$ zu ersetzen ist.

Wir wollen außerdem aussagenlogische Formeln in konjunktiver Normalform (KNF) untersuchen. Eine KNF-Formel ϕ ist eine Konjunktion von Disjunktionstermen, zum Beispiel

$$(x_1 \vee x_2 \vee \neg x_3 \vee x_4) \wedge (x_3 \vee \neg x_2) \wedge (x_1 \vee \neg x_4).$$

Jeder Block von Disjunktionen heißt Klausel und eine Variable oder ihre Negation nennen wir Literal. Ferner bezeichnen wir eine KNF-Formel ϕ mit höchstens 3 Literalen pro Klausel als 3KNF-Formel. Ein Beispiel für eine 3KNF-Formel ist

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4).$$

3.3 NP_{R}-vollständige Probleme über $\mathbb{R}, \mathbb{C}$ und $\mathbb{Z}_2$

Betrachten wir nun die folgenden Probleme[2]:

- SA-FEAS: Gegeben eine semi-algebraische Formel Φ über $\mathbb{R}$, ist Φ erfüllbar?
- QA-FEAS: Gegeben eine quasi-algebraische Formel Φ über R, ist Φ erfüllbar?
- HN: Gegeben eine Menge $f_1, \ldots, f_m$ von Polynomen über R, haben diese eine gemeinsame Nullstelle über R?
- QUAD: Gegeben eine Menge $q_1, \ldots, q_m$ quadratischer Polynome über R, haben diese eine gemeinsame Nullstelle über R?
- 4-FEAS: Gegeben ein Polynom f vom Grad höchstens 4 über R, hat f eine Nullstelle über R?
- 3SAT: Gegeben eine 3KNF-Formel ϕ, ist ϕ erfüllbar?

Dabei betrachten wir SA-FEAS nur über $\mathbb{R}$, da hier auch Ungleichungen der Form $f(x) < 0$ auftauchen können. Das Problem 3SAT untersuchen wir über $\mathbb{Z}_2$, da es hier um aussagenlogische Formeln geht. Wenn wir deutlich machen wollen, dass wir ein Problem PROB nur über einem bestimmten Körper R betrachten, so schreiben wir PROB/R, also zum Beispiel HN /$\mathbb{C}$ wenn wir Systeme polynomieller Gleichungen mit Koeffizienten in $\mathbb{C}$ untersuchen.

Wir bemerken, dass Instanzen von HN, QUAD und 4-FEAS auch Instanzen von QA-FEAS sind, da die Frage, ob die Polynome $f_1, \ldots, f_m$ eine gemeinse Nullstelle haben, äquivalent ist zu der Frage, ob die quasi-algebraische Formel

$$f_1(x) = 0 \ \wedge \cdots \wedge \ f_m(x) = 0$$

erfüllbar ist.

Lassen wir 3SAT zunächst einmal außen vor, so werden die gerade eingeführten Probleme von oben nach unten strukturell einfacher: In einer semi-algebraischen bzw. quasi-algebraischen Formel tauchen Konjunktionen und Disjunktionen polynomieller Gleichungen und Ungleichungen auf, zum Beispiel ist

$$(x > 0 \ \vee \ x^2 + y^2 - 1 = 0) \ \wedge \ (y > 0 \ \vee \ x^2 - 1 = 0)$$

eine semi-algebraische Formel über $\mathbb{R}$. Eine Instanz von HN kann als Konjunktion polynomieller Gleichungen aufgefasst werden, das heißt, Disjunktionen und Ungleichungen fehlen gegenüber SA-FEAS und QA-FEAS:

$$(x^2y^2 - 1 = 0) \ \wedge \ (xyz = 0) \ \wedge \ (x^2 + 1 = 0).$$

[2]Zur Namensgebung: FEAS steht für *feasibility*, was zu Deutsch etwa *Lösbarkeit* bedeutet. Demnach geht es bei SA-FEAS bzw. QA-FEAS um die Frage nach der Lösbarkeit semi-algebraischer (SA) bzw. quasi-algebraischer (QA) Formeln. HN steht für *Hilberts Nullstellensatz*, der die Frage nach der Lösbarkeit polynomieller Gleichungssysteme behandelt (siehe Abschnitt 3). Der Name QUAD verdeutlicht, dass es um Systeme quadratischer Gleichungen geht.

Bei QUAD ist zusätzlich der Grad der auftretenden Polynome durch 2 beschränkt. Bei 4-FEAS betrachten wir statt einem System polynomieller Gleichung nur noch eine einzelne Gleichung vom Grad ≤ 4, ein Beispiel ist

$$2xyz^2 + 4x^2y^2 + 3 = 0.$$

Das nächste Theorem besagt, dass alle Probleme – über dem jeweils passenden Körper – in NP_{R} liegen.

Theorem 3.1. *Es gilt:*

(a) $\mathsf{SA\text{-}FEAS}, \mathsf{QA\text{-}FEAS}, \mathsf{HN}, \mathsf{QUAD}, \mathsf{4\text{-}FEAS} \in \mathsf{NP}_{\mathbb{R}}$.

(b) $\mathsf{QA\text{-}FEAS}, \mathsf{HN}, \mathsf{QUAD}, \mathsf{4\text{-}FEAS} \in \mathsf{NP}_{\mathbb{C}}$.

(c) $\mathsf{QA\text{-}FEAS}, \mathsf{HN}, \mathsf{QUAD}, \mathsf{4\text{-}FEAS}, \mathsf{3SAT} \in \mathsf{NP}_{\mathbb{Z}_2}$.

Beweis. Sei X eines der Probleme über einem passenden Körper. Eine Probleminstanz von X ist entweder eine semi-algebraische bzw. quasi-algebraische Formel Φ, ein System $f_1, \ldots, f_m$ von (quadratischen) Polynomen, ein Polynom f vom Grad ≤ 4 oder eine 3KNF-Formel ϕ. Für die ersten Fälle, bei denen es um polynomiale Gleichungen bzw. Ungleichungen geht, nehmen wir als Zeugen w eine Lösung der Formel Φ bzw. eine gemeinsame Nullstelle der Polynome $f_1, \ldots, f_m$ bzw. eine Nullstelle von f . Es kann dann in polynomieller Zeit von einer BSS-Maschine geprüft werden, ob tatsächlich $\Phi(w) = 1$ bzw. $f_1(w) = \cdots = f_m(w) = 0$ bzw. $f(w) = 0$ gilt. Für die 3KNF-Formel ϕ ist eine erfüllende Belegung v ein geeigneter Zeuge. Auch hier kann eine BSS-Maschine in polynomieller Zeit entscheiden, ob $\phi(v) = 1$ ist. □

Zwei semi-algebraische bzw. quasi-algebraische Formeln Φ und Ψ heißen äquivalent, falls

$$\exists x\ \Phi(x) \quad \Leftrightarrow \quad \exists y\ \Psi(y)$$

gilt. Das heißt, Φ ist genau dann lösbar, wenn Ψ lösbar ist, wobei die Lösungen x und y im Allgemeinen nicht voneinander abhängen.

Theorem 3.2. *Es gilt:*

(a) $\mathsf{SA\text{-}FEAS}, \mathsf{QA\text{-}FEAS}, \mathsf{HN}, \mathsf{QUAD}$ *und* $\mathsf{4\text{-}FEAS}$ *sind* $\mathsf{NP}_{\mathbb{R}}$*-vollständig.*

(b) $\mathsf{QA\text{-}FEAS}, \mathsf{HN}$ *und* QUAD *sind* $\mathsf{NP}_{\mathbb{C}}$*-vollständig.*

(c) $\mathsf{QA\text{-}FEAS}, \mathsf{HN}$ *und* QUAD *sind* $\mathsf{NP}_{\mathbb{Z}_2}$*-vollständig.*

(d) $\mathsf{3SAT}$ *ist* $\mathsf{NP}_{\mathbb{Z}_2}$*-vollständig.*

Bevor wir zum Beweis des Theorems kommen, wollen wir zeigen, dass $\mathsf{4\text{-}FEAS}/\mathbb{C}$ in $\mathsf{P}_{\mathbb{C}}$ liegt: aus dem Fundamentalsatz der Algebra folgt, dass jedes nicht-konstante, komplexe Polynom in n Variablen eine Nullstelle in $\mathbb{C}^n$ hat. Daher kann $\mathsf{4\text{-}FEAS}/\mathbb{C}$ von einer polynomiell zeitbeschränkten Maschine entschieden werden, die genau dann 0 ausgibt, wenn die Eingabe ein von Null verschiedenes, konstantes Polynom ist. Insbesondere folgt daraus, dass $\mathsf{4\text{-}FEAS}/\mathbb{C}$ im Falle $\mathsf{P}_{\mathbb{C}} \neq \mathsf{NP}_{\mathbb{C}}$ nicht $\mathsf{NP}_{\mathbb{C}}$-vollständig ist.

Beweis. Der vollständige Beweis des Theorems ist sehr technisch, daher lassen wir einige Details aus und verweisen auf [1]. Wir zeigen zunächst a), b) und c). In Theorem 3.1 haben wir schon gesehen, dass alle Probleme in NP_{R} für den jeweils passenden Körper R liegen. Es bleibt also noch zu zeigen, dass sich jedes NP_{R}-Problem auf das jeweilige Problem aus dem Theorem reduzieren lässt. Sei also $L \in \mathsf{NP}_{\mathrm{R}}$. Dann gibt es per Definition eine polynomiell zeitbeschränkte Maschine M und ein Polynom q, so dass gilt:

$$x \in L \quad \Leftrightarrow \quad \exists w \in \mathrm{R}^{\leq q(|x|)} \;\; M(x,w) = 1.$$

Im letzten Vortrag wurden die sogenannten Register-Gleichungen $\mathcal{R}$ von M eingeführt und es wurde mit ihrer Hilfe gezeigt, dass es ein Polynom r und eine semi-algebraische (über $\mathbb{R}$) bzw. quasi-algebraische (über $\mathbb{C}$ oder $\mathbb{Z}_2$) Formel Φ gibt, deren Länge polynomiell in $|x|$ ist, so dass

$$\begin{aligned} x \in L \quad &\Leftrightarrow \quad \exists w \in \mathrm{R}^{\leq q(|x|)} \;\; M(x,w) = 1 \\ &\Leftrightarrow \quad \exists w \in \mathrm{R}^{\leq q(|x|)} \;\; \exists u \in \mathrm{R}^{\leq r(|x|)} \;\; \Phi(x,w,u) \end{aligned}$$

gilt. Das bedeutet, dass genau dann x in L liegt, wenn die Formel $\Phi(x,w,u)$ eine Lösung (w,u) hat. Ferner wurde gezeigt, dass $\Phi(x,w,u)$ sogar von einer BSS-Maschine in polynomieller Zeit konstruiert werden kann, so dass die Abbildung $\varphi : x \mapsto \Phi(x,w,u)$ eine polynomielle Reduktion von L auf SA-FEAS bzw. QA-FEAS ist. So konnte die Vollständigkeit von SA-FEAS und QA-FEAS bewiesen werden. Wir wollen jetzt die Formel Φ umformen, und zwar in

- ein äquivalentes System polynomieller Gleichungen,
- ein äquivalentes System quadratischer Gleichungen,
- eine äquivalente polynomielle Gleichung vom Grad ≤ 4 über $\mathbb{R}$.

Da wir auf die genaue Bauart von Φ nicht weiter eingehen wollen, müssen wir an manchen Stellen tiefergehende Untersuchungen auslassen. Nach dem Beweis werden wir die Umformungsschritte noch an einem Beispiel verdeutlichen.

Sei $\Psi = \bigwedge_{j=1}^{m} \phi_j$ eine semi-algebraische Formel mit ϕ_j von der Form

$$f_1(x) < 0 \;\vee \cdots \vee\; f_k(x) < 0 \;\vee\; g_1(x) = 0 \;\vee \cdots \vee\; g_\ell(x) = 0$$

in n Variablen über $\mathbb{R}$. Wir führen für jede Ungleichung $f_j(x) < 0$ eine neue Variable y_j ein und ersetzen

$$f_j(x) < 0 \quad \text{durch} \quad f_j(x) y_j^2 - 1 = 0.$$

Man sieht leicht ein, dass die linke Seite genau dann eine Lösung x hat, wenn die rechte Seite eine Lösung (x, y_j) hat. Daher ist die durch die Ersetzung entstehende quasi-algebraische Formel Ψ' zu Ψ äquivalent. Pro Ungleichung kommt höchstens eine neue Variable hinzu und der Grad der Gleichungen erhöht sich höchstens um 2, so dass die Länge von Ψ' polynomiell in der Länge von Ψ ist.

Sei nun $\Psi' = \bigwedge_{j=1}^{m} \phi'_j$ eine quasi-algebraische Formel mit ϕ'_j von der Form

$$f_1(x) \neq 0 \ \vee \cdots \vee \ f_k(x) \neq 0 \ \vee \ g_1(x) = 0 \ \vee \cdots \vee \ g_\ell(x) = 0$$

in n Variablen $x = x_1, \ldots, x_n$ über R. Um die Ungleichungen zu eliminieren ersetzen wir wie eben

$$f_j(x) \neq 0 \quad \text{durch} \quad f_j(x)y_j - 1 = 0.$$

Das vergrößert die Formel nur polynomiell. Wir haben dann Formeln ϕ'_j der Form

$$h_1(z) = 0 \ \vee \cdots \vee \ h_d(z) = 0,$$

wobei z die Variablen x_i und y_j zusammenfasst und h_j entweder von der Form $f_j(x)y_j - 1$ oder g_j ist. Nun bilden wir für jedes j das Produkt

$$p_j(z) = h_1(z) \cdot \cdots \cdot h_d(z)$$

und erhalten damit das zu Ψ' äquivalente System $\bigwedge_{j=1}^{m}(p_j(x) = 0)$ polynomieller Gleichungen. Für die Register-Gleichungen kann man zeigen, dass das so hergestellte äquivalente System polynomieller Gleichungen auch nur polynomielle Größe in $|x|$ hat.

Sei nun das System polynomieller Gleichungen

$$f_1(x) = 0 \ \wedge \cdots \wedge \ f_m(x) = 0$$

in n Variablen $x = x_1, \ldots, x_n$ über R gegeben. Wir wollen ein äquivalentes, quadratisches System konstruieren. Dazu betrachten wir ein Monom $x_{i_1} \cdots x_{i_k}$ mit mehr als zwei Variablen in einem der f_j und führen neue Variablen $y_{i_1}, \ldots, y_{i_{k-2}}$ ein. Dann ersetzen wir

$$x_{i_1} \cdots x_{i_k} \quad \text{durch} \quad x_{i_1} y_{i_1}$$

und fügen dem System neue Gleichungen

$$\begin{aligned} y_{i_1} - x_{i_2} y_{i_2} &= 0, \\ y_{i_2} - x_{i_3} y_{i_3} &= 0, \\ &\cdots \\ y_{i_{k-2}} - x_{i_{k-1}} x_{i_k} &= 0. \end{aligned}$$

hinzu. Verfahren wir so für jedes Monom, so erhalten wir ein äquivalentes, quadratisches System, so dass die Anzahl der neuen Variablen und Gleichungen durch $K \cdot D$ beschränkt ist, wobei K die Anzahl der Monome und D der maximale Grad der f_j ist. Damit ist die Größe des neuen Systems polynomiell in der Größe des ursprünglichen Systems, und im Falle der Register-Gleichungen polynomiell in $|x|$.

Haben wir ein System quadratischer Gleichungen

$$q_1(x) = 0 \ \wedge \cdots \wedge \ q_m(x) = 0$$

über $\mathbb{R}$ gegeben, so ist

$$\sum_{j=1}^{m} q_j(x)^2 = 0$$

eine äquivalente Gleichung vom Grad höchstens 4. Diese Idee funktioniert übrigens nicht für $\mathbb{C}$ und $\mathbb{Z}_2$, denn die Systeme

$$\begin{aligned} 1 = 0 \ \wedge \ i = 0 \quad &\text{und} \quad 1^2 + i^2 = 0 \text{ über } \mathbb{C} \text{ bzw.} \\ 1 = 0 \ \wedge \ 1 = 0 \quad &\text{und} \quad 1^2 + 1^2 = 0 \text{ über } \mathbb{Z}_2 \end{aligned}$$

sind nicht äquivalent.

Zusammenfassend gilt

$$\begin{aligned} x \in L \quad &\Leftrightarrow \quad \exists w \in \mathrm{R}^{\leq q(|x|)} \ M(x,w) = 1 \\ &\Leftrightarrow \quad \exists w \in \mathrm{R}^{\leq q(|x|)} \ \exists u \in \mathrm{R}^{\leq r(|x|)} \ \Phi(x,w,u) \\ &\Leftrightarrow \quad \exists w \in \mathrm{R}^{\leq q(|x|)} \ \exists v \in \mathrm{R}^{\leq s(|x|)} \ \Psi(x,w,v), \end{aligned}$$

wobei s ein Polynom ist, Φ die zu Beginn des Beweises erwähnte semi-algebraische bzw. quasi-algebraische Formel und Ψ entweder ein System polynomieller Gleichungen über R, ein System quadratischer Gleichungen über R oder eine polynomielle Gleichung vom Grad höchstens 4 über $\mathbb{R}$ ist.

Die oben gezeigten Umformungsschritte können von einer BSS-Maschine durchgeführt werden, so dass die Abbildung $\varphi(x) = \Psi(x,w,v)$ eine polynomielle Reduktion ist. □

Beispiel 3.3. Wir wollen die Umformungsschritte des Beweises anhand der semi - algebraischen Formel

$$(x^2 - x = 0 \ \vee \ x + 1 = 0) \ \wedge \ x > 0$$

über $\mathbb{R}$ verdeutlichen. Die Formel hat die Lösung $x = 1$. Wir führen eine neue Variable y ein und ersetzen die Ungleichung $x > 0$ durch die Gleichung $xy^2 - 1 = 0$. Damit erhalten wir die äquivalente, quasi-algebraische Formel

$$(x^2 - x = 0 \ \vee \ x + 1 = 0) \ \wedge \ xy^2 - 1 = 0$$

in den Variablen x, y. Diese Formel hat die Lösung $(x,y) = (1,1)$. Um zu einem System polynomieller Gleichungen zu gelangen, müssen wir die Disjunktion eliminieren. Dazu bilden wir das Produkt von $x^2 - x$ mit $x+1$ und erhalten die äquivalenten polynomiellen Gleichungen

$$x^3 - x = 0 \ \wedge \ xy^2 - 1 = 0.$$

Auch hier ist $(x,y) = (1,1)$ eine Lösung. Jetzt führen wir für jedes Monom vom Grad größer als 2 eine neue Variable ein, in unserem Fall eine Variable z für das Monom x^3 und w für xy^2. Das obige, polynomielle System ist dann äquivalent zu dem quadratischen System

$$xz - x = 0 \ \wedge \ z - x^2 = 0 \ \wedge \ xw - 1 = 0 \ \wedge \ w - y^2 = 0.$$

Dieses System hat die Lösung $(x, y, z, w) = (1, 1, 1, 1)$. Schließlich bilden wir noch die Summe der Quadrate der Polynome und erhalten die äquivalente Formel vom Grad höchstens 4:

$$x^2z^2 - 4x^2z + x^2 + z^2 + x^4 + x^2w^2 - 2xw + 1 + w^2 - 2wy^2 + y^2 = 0.$$

Auch diese Gleichung hat die Lösung $(x, y, z, w) = (1, 1, 1, 1)$.

Die $\mathsf{NP}_{\mathbb{Z}_2}$-Vollständigkeit von 3SAT folgt mit dem nächsten Theorem und der $\mathsf{NP}_{\mathbb{Z}_2}$-Vollständigkeit von QUAD:

Theorem 3.4. *Es gilt* $\mathsf{QUAD} \leq_p \mathsf{3SAT}$ *über* $\mathbb{Z}_2$.

Beweis. Wir müssen zu jedem quadratischen System $q_1(x) = 0, \ldots, q_m(x) = 0$ über $\mathbb{Z}_2$ eine 3KNF-Formel ϕ polynomieller Länge konstruieren, die genau dann erfüllbar ist, wenn das System eine Lösung hat. Dazu betrachten wir zunächst ein einzelnes quadratisches Polynom q über $\mathbb{Z}_2$ in n Variablen $x_1, \ldots, x_n$ und schreiben q als

$$q(x) = \sum_{i=1}^{s} \sigma_i + \sum_{j=1}^{t} \tau_j + e, \tag{1}$$

wobei σ_i die Form $x_k x_\ell$ und τ_j die Form x_k hat und $e \in \mathbb{Z}_2$ ist. In der ersten Summe stehen also alle Monome mit zwei Variablen, in der zweiten Summe alle Monome mit einer Variable. Nun ist q äquivalent zu folgendem System Φ in den Variablen $x_1, \ldots, x_n$ und den neuen Variablen $S, T, \sigma_i, \tau_j, y_k, w_\ell$:

$$S + T + e = 0 \tag{2}$$

$$\sigma_i + x_k x_\ell = 0,\ (i = 1, \ldots, s), \quad \tau_j + x_k = 0,\ (j = 1, \ldots, t) \tag{3}$$

$$S + \sigma_1 + y_1 = 0, \quad y_1 + \sigma_2 + y_2 = 0,\ \ldots\ , y_{s-2} + \sigma_{s-1} + \sigma_s = 0 \tag{4}$$

$$T + \tau_1 + w_1 = 0, \quad w_1 + \tau_2 + w_2 = 0,\ \ldots\ , w_{t-2} + \tau_{t-1} + \tau_t = 0 \tag{5}$$

Beachte dazu, dass $a = -a$ für $a \in \mathbb{Z}_2$ gilt, und dass S für $\sum_{i=1}^{s} \sigma_i$ und T für $\sum_{j=1}^{t} \tau_j$ steht. Das System Φ besteht aus $2(s + t) - 1$ Gleichungen und hat $2(s + t) - 2$ neue Variablen. Da q genau $s + t + 1$ Summanden hat, ist die Länge von Φ polynomiell in der Länge von q. Im Folgenden bezeichnen wir die Variablen von Φ einheitlich mit u_j.

Wir bemerken, dass alle Gleichungen des Systems Φ von der Form

$$a + b + c = 0 \qquad \text{oder} \qquad a + bc = 0$$

sind, wobei a, b, c entweder Variablen u_j oder 0 oder 1 sind. Die Gleichung $a + b + c = 0$ ist äquivalent zu dem System

$$abc = 0, \quad a(b + 1)(c + 1) = 0, \quad (a + 1)b(c + 1) = 0, \quad (a + 1)(b + 1)c = 0, \tag{6}$$

und $a + bc$ ist äquivalent zu dem System

$$a(b + 1) = 0, \quad a(c + 1) = 0, \quad (a + 1)bc = 0. \tag{7}$$

Das kann man leicht nachprüfen, indem man alle 0-1-Belegungen für a, b und c durchgeht. Ersetzt man nun in Φ jede Gleichung durch das entsprechende System von höchstens 4 Gleichungen gemäß (6) bzw. (7), so erhält man ein zu q äquivalentes System Φ' mit höchstens $4(s+t)$ Gleichungen in den selben Variablen wie Φ, so dass jede Gleichung in Φ' die Form $z_1 z_2 z_3 = 0$ hat, wobei z_k entweder u_j, $u_j + 1$ oder 1 ist. Wir ordnen einer Gleichung $z_1 z_2 z_3 = 0$ die Formel $v_1 \vee v_2 \vee v_3$ zu, wobei

$$v_k = \begin{cases} u_j, & \text{falls } z_k = u_j + 1 \\ \neg u_j, & \text{falls } z_k = u_j \\ 0, & \text{falls } z_k = 1 \end{cases}$$

so dass genau dann $z_1 z_2 z_3 = 0$ gilt wenn $v_1 \vee v_2 \vee v_3 = 1$ ist. Das machen wir für jede Gleichung von Φ' und erhalten eine 3KNF Formel ϕ_q polynomieller Länge mit

$$\exists x\ q(x) = 0 \qquad \Leftrightarrow \qquad \exists b\ \phi_q(b) = 1.$$

Haben wir nun ein quadratisches System $q_1(x) = 0, \ldots, q_m(x) = 0$, so bilden wir die 3KNF-Formel $\phi \equiv \phi_{q_1} \wedge \cdots \wedge \phi_{q_m}$. Man macht sich leicht klar, dass die Konstruktion von ϕ von einer BSS-Maschine durchgeführt werden kann. Zusammenfassend gilt dann

$$\exists x\ q_1(x) = 0\ \wedge \cdots \wedge\ q_m(x) = 0 \qquad \Leftrightarrow \qquad \exists b\ \phi(b) = 1,$$

das heißt wir haben QUAD auf 3SAT reduziert. □

Beispiel 3.5. Betrachte die quadratische Gleichung

$$q(x) = x_1 x_2 + x_2 x_3 + x_1 + 1 = 0.$$

Wir wollen eine zu $q(x) = 0$ äquivalente 3KNF-Formel ϕ konstruieren. Mit den Bezeichnungen des vorangegangenen Beweises ist

$$s = 2,\ t = 1,\ \sigma_1 = x_1 x_2,\ \sigma_2 = x_2 x_3,\ \tau_1 = x_1,\ e = 1,\ S = x_1 x_2 + x_2 x_3,\ T = x_1.$$

Im ersten Schritt stellen wir das System Φ in den Variablen $x_1, x_2, x_3, S, T, \sigma_1, \sigma_2, \tau_1$ auf:

$$\begin{gathered} S + T + 1 = 0, \\ \sigma_1 + x_1 x_2 = 0, \quad \sigma_2 + x_2 x_3 = 0, \quad \tau_1 + x_1 = 0, \\ S + \sigma_1 + \sigma_2 = 0, \\ T + \tau_1 = 0. \end{gathered}$$

Nun ersetzen wir jede der Gleichungen durch höchstens 4 neue Gleichungen und erhalten das äquivalente System Φ':

$$\begin{gathered} ST = 0, \quad (S+1)(T+1) = 0, \\ \sigma_1(x_1+1) = 0, \quad \sigma_1(x_2+1) = 0, \quad (\sigma_1+1)x_1x_2 = 0, \\ \sigma_2(x_2+1) = 0, \quad \sigma_2(x_3+1) = 0, \quad (\sigma_2+1)x_2x_3 = 0, \\ \tau_1(x_1+1) = 0, \quad (\tau_1+1)x_1 = 0, \\ S\sigma_1\sigma_2 = 0, \quad S(\sigma_1+1)(\sigma_2+1) = 0, \quad (S+1)\sigma_1(\sigma_2+1) = 0, \quad (S+1)(\sigma_1+1)\sigma_2 = 0, \\ T(\tau_1+1) = 0, \quad (T+1)\tau_1 = 0. \end{gathered}$$

Dabei sind einige Gleichungen weggefallen, wenn $c = 1$ oder $c = 0$ war. Wir bilden beispielhaft 3 der 16 zugehörigen Klauseln:

$$\begin{aligned} ST = 0 \quad &\rightarrow \quad (\neg S) \vee (\neg T) \vee 0, \\ (S+1)(T+1) = 0 \quad &\rightarrow \quad S \vee T \vee 0, \\ (\sigma_2 + 1)x_2x_3 = 0 \quad &\rightarrow \quad \sigma_2 \vee (\neg x_2) \vee (\neg x_3). \end{aligned}$$

Man erhält schließlich eine zu $q(x) = 0$ äquivalente 3KNF-Formel ϕ, indem man für alle 16 Gleichungen des Systems Φ' die zugehörigen Klauseln ausrechnet und ihre Konjunktion bildet.

4 Zur Entscheidbarkeit von NP_{R} über $\mathbb{R}, \mathbb{C}$ und $\mathbb{Z}_2$

In diesem Abschnitt wollen wir die Beweisideen für das folgende Theorem darlegen:

Theorem 4.1. *Die Probleme* 4-FEAS $/\mathbb{R}$, HN $/\mathbb{C}$ *und* 3SAT $/\mathbb{Z}_2$ *sind entscheidbar.*

Dabei ist die Entscheidbarkeit von 3SAT leicht einzusehen: Das Entscheidungsverfahren geht für eine Formel φ in n Variablen alle möglichen Belegungen $x \in \{0,1\}^n$ durch und prüft jeweils den Wahrheitswert von $\varphi(x)$. Falls φ einmal erfüllt ist, gibt das Verfahren 1 aus, ansonsten 0. Da die Länge von φ im Wesentlichen gleich n ist (die Anzahl der Variablen plus höchstens eine Negation $\neg$ und ein Konnektor $\vee$ oder $\wedge$ pro Variable) und der Wert von $\varphi(x)$ für jede der 2^n vielen Belegungen $x \in \{0,1\}^n$ in polynomieller Zeit überprüfbar ist, hat das Verfahren höchstens exponentielle Laufzeit. Mit der $\mathsf{NP}_{\mathbb{Z}_2}$-Vollständigkeit von 3SAT folgt außerdem $\mathsf{NP}_{\mathbb{Z}_2} \subseteq \mathsf{EXP}_{\mathbb{Z}_2}$.

Um zu zeigen, dass 4-FEAS $/\mathbb{R}$ und HN $/\mathbb{C}$ entscheidbar sind, benötigen wir tiefer liegende Resultate der Logik und der algebraischen Geometrie, die wir ohne Beweise angeben werden. Zunächst noch eine einfache, aber wichtige Folgerung aus Theorem 4.1 und der NP_{R}-Vollständigkeit der dort genannten Probleme:

Korollar 4.2. *Für* $\mathrm{R} \in \{\mathbb{R}, \mathbb{C}, \mathbb{Z}_2\}$ *gilt: Alle Probleme in* NP_{R} *sind entscheidbar.*

Für andere Ringe gilt die Aussage des Korollars im Allgemeinen nicht: So kann man zum Beispiel zeigen, dass $\mathsf{NP}_{\mathbb{Q}}$ und $\mathsf{NP}_{\mathbb{Z}}$ unentscheidbare Probleme enthalten. Siehe dazu [1].

4.1 Ein Entscheidungsverfahren für 4-FEAS über $\mathbb{R}$

Es soll nun ein Verfahren konstruiert werden, das entscheidet, ob ein reelles Polynom f in mehreren Variablen vom Grad kleiner oder gleich 4 eine reelle Nullstelle hat. Anders ausgedrückt muss das Verfahren entscheiden, ob für ein gegebenes Polynom f der Satz

$$\exists x_1 \ldots \exists x_n \; f(x_1, \ldots, x_n) = 0$$

wahr ist. Wir betrachten zunächst allgemeiner quantifizierte Formeln der Gestalt

$$\varphi(z_1, \ldots, z_\ell) \quad \equiv \quad Q_1 x_1 \ldots Q_n x_n \; \Phi(x_1, \ldots, x_n, z_1, \ldots, z_\ell) \qquad (*)$$

mit freien reellen Variablen $z_1, \ldots, z_\ell$ und Quantoren $Q_i \in \{\forall, \exists\}$, wobei Φ eine semialgebraische Formel über $\mathbb{R}$ ist. Um den Wahrheitswert einer solchen Formel zu bestimmen, wollen wir sie zuerst vereinfachen, indem wir die Quantoren aus der Formel eliminieren. Damit ist gemeint, dass wir eine zu φ äquivalente Formel ψ ohne Quantoren suchen. Zum Beispiel folgt mithilfe der Lösungsformel für quadratische Gleichungen, dass die Formel

$$\varphi(a, b, c) \;\equiv\; \exists x\; ax^2 + bx + c = 0$$

mit den freien reellen Variablen a, b, c äquivalent ist zur quantorenfreien Formel

$$\psi(a, b, c) \;\equiv\; b^2 - 4ac \geq 0.$$

Der folgende Satz von Tarski [7], der sich in dieser Fassung in [6] findet, besagt nun, dass eine solche *Quantorenelimination* über $\mathbb{R}$ für Formeln der Gestalt $(*)$ stets möglich und sogar berechenbar ist. In der Seminarausarbeitung [5] wird der Satz ausführlich bewiesen und anhand einiger Beispiele veranschaulicht.

Theorem 4.3 (Tarski)**.** *Der Körper der reellen Zahlen* $\mathbb{R}$ *erlaubt* effektive Quantorenelimination*, das heißt, zu jeder* quantifizierten *Formel*

$$\varphi(z_1, \ldots, z_\ell) \quad \equiv \quad Q_1 x_1 \ldots Q_n x_n\; \Phi(x_1, \ldots, x_n, z_1, \ldots, z_\ell)$$

der Form $(*)$ *gibt es eine äquivalente,* quantorenfreie *Formel* $\psi(z_1, \ldots, z_\ell)$ *mit denselben freien Variablen und der gleichen Bauart, und das Eliminationsverfahren ist von einer BSS-Maschine berechenbar.*

Das Theorem von Tarski liefert nun folgendes Entscheidungsverfahren für 4-FEAS: Bei Eingabe eines reellen Polynoms f vom Grad ≤ 4 in n Variablen wendet das Verfahren die Quantorenelimination auf den Satz $\exists x_1 \ldots \exists x_n\; f(x_1, \ldots, x_n) = 0$ an und prüft dann, ob der äquivalente, quantorenfreie Satz wahr ist. Diese Überprüfung kann von einer BSS-Maschine durchgeführt werden, da das Ergebnis der Quantorenelimination keine freien Variablen enthält und daher eine boolesche Kombination von Gleichungen und Ungleichungen bestehend aus Summen und Produkten reeller Zahlen ist.

Übergibt man dem Verfahren zum Beispiel das Polynom $3x^2 - \pi x + \sqrt{2}$, so bildet es den Satz $\exists x\; 3x^2 - \pi x + \sqrt{2} = 0$ und ruft damit den Algorithmus zur Quantorenelimination auf. Das Ergebnis könnte etwa der äquivalente Satz $(-\pi)^2 - 4 \cdot 3 \cdot \sqrt{2} \geq 0$ sein (wobei das genaue Ergebnis vom gewählten Eliminationsverfahren abhängt), der nun leicht zu überprüfen ist.

Das erste, von Tarski in [7] vorgestellte Verfahren zur Quantorenelimination ist nicht praktisch verwendbar: seine Laufzeit für eine Formel mit n Variablen ist etwa

$$2^{2^{2^{\cdots^2}}} \Big\} n\text{-mal},$$

also ein Turm von n Zweien. Ein 1988 in [4] vorgeschlagener Algorithmus erlaubt Quantorenelimination in einer Laufzeit, die exponentiell in der Anzahl n der Variablen und doppelt exponentiell in der Anzahl der Quantorenalternationen ist. Für unsere Formeln, die

nur Existenzquantoren enthalten, hat der Algorithmus also exponentielle Laufzeit. Damit folgt insbesondere $\mathsf{4\text{-}FEAS} \in \mathsf{EXP}_{\mathbb{R}}$ und wegen der $\mathsf{NP}_{\mathbb{R}}$-Vollständigkeit von $\mathsf{4\text{-}FEAS}$ sogar $\mathsf{NP}_{\mathbb{R}} \subseteq \mathsf{EXP}_{\mathbb{R}}$.

4.2 Ein Entscheidungsverfahren für HN über $\mathbb{C}$

Das Problem, zu entscheiden, ob eine Menge $f_1, \ldots, f_m$ von komplexen Polynomen eine gemeinsame Nullstelle hat, wird unter Anderem in der Algebraischen Geometrie untersucht. Der Hilbertsche Nullstellensatz liefert ein Kriterium, wann es eine solche gemeinsame Nullstelle gibt:

Theorem 4.4 (Hilbertscher Nullstellensatz)**.** *Seien $f_1, \ldots, f_m \in \mathbb{C}[X_1, \ldots, X_n]$ Polynome. Die f_j haben genau dann* keine *gemeinsame Nullstelle, wenn es Polynome $g_1, \ldots, g_m \in \mathbb{C}[X_1, \ldots, X_n]$ gibt mit*

$$\sum_{j=0}^{m} g_j f_j = 1.$$

Dabei können die g_j so gewählt werden, dass $\operatorname{grad} g_j \leq D^n$ *gilt, wobei*

$$D = \max\{3, \operatorname{grad}(f_1), \ldots, \operatorname{grad}(f_m)\}.$$

Diese Version des Satzes, in der eine Gradschranke für die g_j angegeben wird, stammt aus [1] und wird auch als effektiver Hilbertscher Nullstellensatz bezeichnet.

Beispiel 4.5. Betrachten wir ein Beispiel für die Anwendung des Satzes: Die Polynome

$$f_1(x, y) = x^2 + xy + 1 \quad \text{und} \quad f_2(x, y) = x + y$$

haben nach dem Hilbertschen Nullstellensatz keine gemeinsame Nullstelle, da $1 \cdot f_1 + (-x) \cdot f_2 = 1$ ist. Dagegen besitzen die Polynome

$$f_3(x, y) = x^2 + y^2 \quad \text{und} \quad f_4(x, y) = x + y$$

offenbar die gemeinsame Nullstelle $(0, 0)$, das heißt es gibt nach dem Satz keine Polynome $g_3, g_4 \in \mathbb{C}[x, y]$ mit $f_3 g_3 + f_4 g_4 = 1$.

Ein mögliches Entscheidungsverfahren für HN über $\mathbb{C}$ sieht nun wie folgt aus: Bei Eingabe von Polynomen $f_1, \ldots, f_m \in \mathbb{C}[X_1, \ldots, X_n]$:

1. Berechne die Schranke $d := D^n$ für die Grade der g_j.
2. Mache für $j = 1, \ldots, m$ den Ansatz $g_j = \sum_{|\alpha| \leq d} c_{\alpha j} X^\alpha$ mit Variablen $c_{\alpha j}$.
3. Überprüfe (z.B. mittels Gauß-Elimination), ob das lineare System $\sum_{j=1}^{m} f_j g_j = 1$ in den Variablen $c_{\alpha j}$ lösbar ist.
4. Ist das System lösbar, so gibt 1 aus. Andernfalls, gib 0 aus.

Der Algorithmus liefert auch die Koeffizienten der g_j, falls die $f_1, \ldots, f_m$ keine gemeinsame Nullstelle haben. Wir wollen die Funktionsweise des Verfahrens an einem Beispiel vorführen:

Beispiel 4.6. Betrachte die Polynome $f_1, f_2 \in \mathbb{C}[x, y]$ mit

$$f_1(x, y) = xy + 1, \quad f_2(x, y) = y.$$

Anwenden des Hilbertschen Nullstellensatz mit $g_1(x, y) = 1$ und $g_2(x, y) = -x$ zeigt, dass f_1 und f_2 keine gemeinsame Nullstelle haben. Der Algorithmus sollte also auch zu diesem Ergebnis kommen. Er würde zunächst die Gradschranke $d := D^n$ bestimmen, in unserem Fall $d = 9$. Da wir aber schon wissen, wie g_1, g_2 gewählt werden können, nehmen wir der Einfachheit halber $d = 1$. Mache den Ansatz

$$\begin{aligned} g_1 &= \sum_{|\alpha| \leq 1} b_\alpha X^\alpha = b_{00} + b_{01}y + b_{10}x, \\ g_2 &= \sum_{|\alpha| \leq 1} c_\alpha X^\alpha = c_{00} + c_{01}y + c_{10}x. \end{aligned}$$

Gesucht sind nun die b_α, c_α, so dass $g_1 f_1 + g_2 f_2 = 1$. Es ist

$$\begin{aligned} g_1 f_1 + g_2 f_2 &= (b_{00}xy + b_{00} + b_{01}xy^2 + b_{01}y + b_{10}x^2y + b_{10}x) + (c_{00}y + c_{01}y^2 + c_{10}xy) \\ &= b_{00} + (b_{01} + c_{00})y + c_{01}y^2 + b_{10}x + (b_{00} + c_{10})xy + b_{01}xy^2 + b_{10}x^2y. \end{aligned}$$

Löse nun (im Allgemeinen mit Gaußeliminiation) $g_1 f_1 + g_2 f_2 = 1$, also:

$$b_{00} = 1,\ b_{01} + c_{00} = 0,\ c_{01} = 0,\ b_{10} = 0,\ b_{00} + c_{10} = 0,\ b_{01} = 0,\ b_{10} = 0.$$

Das liefert $g_1 = 1,\ g_2 = -x$.

Man kann zeigen, dass die Anzahl der Koeffizienten der g_j exponentiell in der Eingabegröße wächst, siehe [1]. Da die Gaußelimination in polynomieller Zeit durchführbar ist, folgt $\mathsf{HN} \in \mathsf{EXP}_{\mathbb{C}}$ und damit sogar $\mathsf{NP}_{\mathbb{C}} \subseteq \mathsf{EXP}_{\mathbb{C}}$.

Literatur

[1] Blum, L., Cucker, F., Shub, M., and Smale, S. *Complexity and Real Computation.* Springer, 1998.

[2] Blum, L., Shub, M., and Smale, S. *On a Theory of Computation and Complexity over the Real Numbers: NP-Completeness, Recursive Functions and Universal Machines.* Bulletin (New Series) of the American Mathematical Society, Vol. 21, No 1., 1989.

[3] Cucker, F. *On the Complexity of Quantifier Elimination: the Structural Approach.* The Computer Journal, Vol. 36, No. 5, 1993.

[4] Grigor'ev, D. Y. *The Complexity of deciding Tarski Algebra.* Journal of Symbolic Computation, 1988.

[5] Günther, F. Algorithmische Aspekte der Quantorenelimination auf den reellen Zahlen. Seminarausarbeitung, `https://www3.mathematik.tu-darmstadt.de/index.php?id=84&evsid=32&evsver=927&evsdir=965&evsfile=Guenther.pdf`, 2011.

[6] Makowski, J. Existential Theory of the Real Numbers. `www.cs.technion.ac.il/~janos/COURSES/THPR/tarskiproof.ps`.

[7] Tarski, A. *A Decision Method for Elementary Algebra and Geometry.* RAND Corporation, 1951.