Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Informatik I

# Komplexität von Constraint Satisfaction Problemen über endlichen Teilmengen natürlicher Zahlen

## Master-Thesis

Eingereicht am 21.12.2015

überarbeitete Version vom 24.02.2016

von Titus Dose

Betreuer und Erstgutachter: Prof. Dr. Christian Glaßer
Zweitgutachter: Prof. Dr. Klaus W. Wagner

# Zusammenfassung

Die vorliegende Arbeit ist der Betrachtung zweier spezieller Sorten von Constraint Satisfaction Problemen gewidmet: Die Domäne der Variablen und Konstanten ist jeweils auf endliche Teilmengen der natürlichen Zahlen eingeschränkt. Constraints sind Gleichungen von Termen aus Variablen und Konstanten über einer Menge von Operationen. Die zugelassenen Operationen sind dabei Teilmengen der folgenden fünf Operationen: Die drei Mengenoperationen Vereinigung, Schnitt und Mengendifferenz sowie die paarweise Addition und paarweise Multiplikation.

1. In der ersten Variante umfasst die Domäne der Variablen und Konstanten alle endlichen Teilmengen der natürlichen Zahlen.

2. Bei der zweiten Variante ist die Domäne auf endliche Intervalle über den natürlichen Zahlen eingeschränkt.

Lässt man an Operationen nur eine Auswahl der Mengenoperationen zu, so zeigen wir für jedes dieser Probleme die $\leq_{\mathrm{m}}^{\log}$-Vollständigkeit für eine der Komplexitätsklassen L, P und NP. In diesem Bereich gibt es kaum Unterschiede zwischen den beiden von uns betrachteten Problemvarianten.

Beim Vorliegen von arithmetischen Operationen bleiben einige offene Fragen. Mit den einzelnen Problemen wird eine wesentlich weitere Spanne von Komplexitätsklassen erreicht. So zeigen wir für einzelne Probleme die NP-Vollständigkeit, für andere Probleme die $\Sigma_1$-Vollständigkeit.
Dabei sind die von uns gezeigten oberen Komplexitätsschranken für die Variante über endlichen Intervallen zum Teil deutlich kleiner als jene für die Variante über beliebigen endlichen Teilmengen der natürlichen Zahlen.

# Inhaltsverzeichnis

# 1 Einleitung

Ein Constraint Satisfaction Problem (CSP) ist ein Entscheidungsproblem, bei dem eine endliche Menge von Variablen und eine endliche Menge von Constraints als Eingabe übergeben werden und dann zu entscheiden ist, ob die Variablen so mit Werten aus einer gegebenen Domäne belegt werden können, dass alle Constraints erfüllt sind.

Beim Vorliegen einer endlichen Domäne und beliebigen Constraints erweist sich das jeweilige CSP als NP-vollständig. Lässt man in den Constraints jedoch nur eine eingeschränkte Auswahl an Relationen zu, so ergeben sich auch Probleme in P.

Die Menge der Relationen, die in den Constraints erlaubt sind, wird Constraintsprache genannt. Die Frage, für welche Constraintsprachen die zugehörigen CSPs in Polynomialzeit entschieden werden können, hat sich in den vergangenen Jahren zu einem Gebiet ausgiebiger Forschung entwickelt.
Feder und Vardi [FV99] vermuten eine Dichotomie für CSPs über endlichen Domänen derart, dass solche CSPs entweder in P oder aber NP-vollständig sind. Diese Vermutung konnte jedoch bislang nicht bewiesen werden.

Ferner beginnt sich ein steigendes Interesse an CSPs über unendlichen Domänen zu entwickeln. Hier ergeben sich zum Teil wesentlich höhere Komplexitäten. So wird auch in dieser Arbeit die Unentscheidbarkeit für eine Reihe von Problemen bewiesen. Es ergeben sich aber auch sehr einfache Probleme, die etwa L- oder P-vollständig sind. Insgesamt wird so eine sehr große Spanne an Komplexitätsklassen abgedeckt.

Ein weiterer, stärker theoretisch motivierter Bereich, aus dem heraus sich die in dieser Arbeit untersuchten Fragen entwickelt haben, sind arithmetische Formeln und Schaltkreise.

Die Frage nach der Komplexität von Entscheidungsproblemen bezüglich derartiger Formeln oder Schaltkreise hat seinen Ursprung bei Meyer und Stockmeyer [SM73]. Diese untersuchen sogenannte Integer-Ausdrücke. Ein Integer-Ausdruck ist ein Term aus Konstanten, welche einelementige Teilmengen der natürlichen Zahlen sind, sowie der paarweisen Addition zweier Mengen und Mengenoperationen wie Vereinigung oder Schnitt. Meyer und Stockmeyer untersuchten das Membership-Problem, also die Frage, ob eine gegebene natürliche Zahl in der von einem Integer-Ausdruck beschriebenen Teilmenge von $\mathbb{N}$ enthalten ist, sowie das Inäquivalenz-Problem, also die Frage, ob zwei gegebene Integer-Ausdrücke verschiedene Mengen beschreiben.

Aufbauend auf dieser und anderen Arbeiten untersuchten McKenzie und Wagner [MW03] die Komplexität von Membership-Problemen von Schaltkreisen über Mengen natürlicher Zahlen. Die Fragestellung war eine ähnliche: Ist eine gegebene natürliche Zahl Element der von einem Schaltkreis berechneten Teilmenge von $\mathbb{N}$?
Ein Schaltkreis ist dabei ein gerichteter, azyklischer Graph mit zwei Sorten von Knoten: Zum einen gibt es Eingabeknoten, welchen eine natürliche Zahl zugewiesen wird, alle anderen Knoten, im Folgenden Operationsknoten genannt, wird eine Operation zugewiesen. An Operationen können die paarweise Addition und Multiplikation, sowie Mengenoperationen verwendet werden. Die Anzahl der eingehenden Kanten ist für einen Operationsknoten durch die Stelligkeit der ihm zugewiesenen Operation gegeben.
Es gibt in einem Schaltkreis weiterhin einen ausgezeichneten Knoten: Den Ausgabeknoten. Jeder Knoten berechnet eine Menge natürlicher Zahlen: Während jeder Eingabeknoten die einelementige Menge berechnet, die ihm zugewiesen ist, berechnet jeder Operationsknoten

das Ergebnis der Anwendung der ihm zugewiesenen Operation auf die von den Vorgängerknoten berechneten Mengen. Der Schaltkreis berechnet insgesamt die vom Ausgabeknoten berechnete Menge.

Im Gegensatz zu den Integer-Ausdrücken ist es diesen Schaltkreisen möglich, Zwischenergebnisse zwischenzuspeichern und an verschiedenen Stellen wiederzuverwenden. Schaltkreise bieten also die Möglichkeit große Zahlen und Mengen in knapperer Form zu beschreiben.

Glaßer et al. [GHR$^+$07] untersuchten für die beschriebenen Schaltkreise, ähnlich wie Meyer und Stockmeyer für Integer-Expressions, das Äquivalenz-Problem.

In einer weiteren Arbeit untersuchten Glaßer et al. [GRTW10] das Erfüllbarkeitsproblem für Schaltkreise. Die untersuchten Schaltkreise können auch unbelegte Eingangsknoten enthalten. Das Problem war nunmehr, für eine gegebene natürliche Zahl $b$ zu entscheiden, ob die unbelegten Eingangsknoten mit einelementigen Teilmengen der natürlichen Zahlen belegt werden können, sodass $b$ Element der vom Schaltkreis berechneten Menge ist.
Durch diese Modifikation sind die entstehenden Schaltkreisen den CSPs noch ähnlicher.

Glaßer et al. [GJM15] brachten nun die CSPs und die Schaltkreise zusammen. Die von ihnen untersuchten CSPs besaßen die Menge aller einelementigen Teilmengen von $\mathbb{N}$ als Domäne und erlaubten als Operationen Teilmengen von $\{+, \times, \cup, \cap, ^-\}$.
Jedoch ist die Menge der einelementigen Teilmengen von $\mathbb{N}$ nicht abgeschlossen bezüglich der genannten Mengenoperation. Deshalb ist es natürlich, CSPs mit endlichen Teilmengen der natürlichen Zahlen als Domäne zu betrachten. An dieser Stelle setzt die vorliegende Arbeit an. Es können einige Resultate aus den genannten Arbeiten übertragen werden, jedoch stellt sich auch heraus, dass einige Fragestellungen durch die vergrößerte Domäne deutlich schwieriger werden.
Als weiteren Aspekt dieser Arbeit betrachten wir CSPs über der Domäne der endlichen Intervalle von natürlichen Zahlen. Dadurch lassen sich zum Teil leichter Vollständigkeiten für verschiedene Komplexitätsklassen zeigen.

# 2 Grundlagen

Wir beginnen mit ein paar kurzen Bemerkungen zu den in dieser Arbeit verwendeten Notationen und Codierungen.

## 2.1 Grundlegende Notationen und Codierungen von Objekten

Wir verwenden Standardnotationen:
Es bezeichne $\mathbb{N}$ die Menge der natürlichen Zahlen $\{0, 1, 2, \dots\}$ und $\mathbb{N}^+$ die Menge der positiven natürlichen Zahlen $\{1, 2, 3, \dots\}$.
Mit $\mathcal{P}_{\text{fin}}(\mathbb{N})$ sei die Menge aller endlichen Teilmengen von $\mathbb{N}$ bezeichnet. Es gilt also

$$\mathcal{P}_{\text{fin}}(\mathbb{N}) =_{\text{def}} \{A \mid A \subseteq \mathbb{N}\}.$$

Weiterhin sei $[\mathbb{N}]$ die Menge aller endlichen Intervalle über den natürlichen Zahlen, also

$$[\mathbb{N}] =_{\text{def}} \{\{x \mid a \leq x \leq b\} \mid a, b \in \mathbb{N}\}.$$

Dabei werde ein konkretes Intervall $\{x \mid a \leq x \leq b\}$ für natürliche Zahlen $a, b$ mit $[a, b]$ bezeichnet. Die Menge der Randpunkte $\{a, b\}$ eines Intervals $[a, b]$ bezeichnen wir mit $R([a, b])$. Es sei insbesondere darauf hingewiesen, dass $[a, b] = \emptyset$, falls $a > b$ gilt.
Des Weiteren sei

$$\{\mathbb{N}\} =_{\text{def}} \{\{n\} \mid n \in \mathbb{N}\}$$

die Menge der einelementigen Teilmengen von $\mathbb{N}$.

Außerdem verwenden wir die folgenden verbreiteten Notationen für Komplexitätsklassen.

| **Komplexitätsklasse** | **enthält die Probleme, entscheidbar in** |
|---|---|
| L | deterministischem logarithmischem Raum |
| NL | nichtdeterministischem logarithmischem Raum |
| P | deterministischer Polynomialzeit |
| NP | nichtdeterministischer Polynomialzeit |
| PSPACE | deterministischem polynomiellem Raum |
| EXP | deterministischer Exponentialzeit |
| NEXP | nichtdeterministischer Exponentialzeit |

Ferner bezeichne $\Sigma_1$ die Menge der aufzählbaren Probleme. Wir verwenden zudem die üblichen Bezeichnungen $\Sigma_i^{\text{p}}$ und $\Pi_i^{\text{p}}$ für die Klassen der Polynomialzeit-Hierarchie, wobei $i \in \mathbb{N}$ sei.

Des weiteren schreiben wir für $A, B \subseteq \mathbb{N}$

$$A \leq_{\text{m}}^{\log} B \quad \text{bzw.} \quad A \leq_{\text{m}}^{\text{p}} B,$$

wenn es eine in logarithmischem Raum bzw. polynomieller Zeit berechenbare, totale Funktion $f : \mathbb{N} \to \mathbb{N}$ gibt, sodass $x \in A$ genau dann gilt, wenn $f(x) \in B$.

Es sei an dieser Stelle darauf hingewiesen, dass wir für gängige NP-vollständige Probleme wie SAT, 3SAT oder SOS das Wissen, dass diese sogar $\leq_{\text{m}}^{\log}$-vollständig für NP sind, als bekannt voraussetzen.

Bei Funktionen gehen wir im Allgemeinen von totalen Funktionen aus, ohne dies jedes Mal explizit zu erwähnen. Wir weisen zur Erinnerung dennoch immer wieder bei einzelnen Funktionen darauf hin, dass diese total sind.

Mit $W_f$ für eine Funktion $f : A \to B$ und beliebige Mengen $A, B$ notieren wir die Menge $f(A) = \{f(x) \mid x \in A\}$.

Zudem bemerken wir einige Aspekte zur Codierung von Objekten durch natürliche Zahlen bzw. ein Wort über einem endlichen Alphabet:
In dieser Arbeit werden ausschließlich Standardcodierungen verwendet. Die genaue Codierung ist dabei in aller Regel nicht von Relevanz. Wir gehen davon aus, dass natürliche Zahlen stets binär, dyadisch oder $k$-adisch für ein $k > 2$ dargestellt sind. In jedem Fall ist die Länge der Codierung einer natürlichen Zahl $n$ in $\Theta(\log n)$.
Für die Länge einer Codierung eines beliebigen Objektes $o$ schreiben wir $|o|$. Bei endlichen Mengen $M$ bezeichnen wir jedoch mit $|M|$ die Kardinalität der Menge $M$ und nicht die Länge der Codierung.

Diese Arbeit befasst sich mit zwei Varianten von sogenannten Constraint Satisfaction Problemen: In der einen Variante spielen beliebige endliche Teilmengen der natürlichen Zahlen eine wesentliche Rolle, während in der anderen Variante nur endliche Intervalle von natürlichen Zahlen zugelassen sind.
Bei Problemen der ersten Variante gehen wir von einer Codierung aus, sodass endliche Teilmengen der natürlichen Zahlen $\{a_1, \ldots, a_k\}$ für $k \in \mathbb{N}^+$ so codiert sind, dass die Länge der Codierung in $\Theta(\sum_{i=1}^{k} |a_k|)$ ist.
Im zweiten Fall sei ein Intervall $[a, b]$ für natürliche Zahlen $a < b$ so codiert, dass die Länge der Codierung in $\Theta(|a| + |b|)$ ist.

Bei Graphen gehen wir von Adjazenzmatrizen als Repräsentationsform aus.

Zuletzt sei noch bemerkt, dass wir alle in dieser Arbeit betrachteten Probleme als Teilmengen natürlicher Zahlen auffassen, auch wenn wir der Einfachheit halber einzelne Probleme beispielsweise als Mengen von Formeln oder Graphen definieren.

## 2.2 Definition der zentralen Probleme

Wir werden im Rahmen dieser Arbeit zwei spezielle Sorten von CSPs betrachten und diese im Folgenden präzise definieren.

**Definition 2.1**
Wir definieren zunächst **Terme**. Es sei $X$ ein Variablen- oder Konstantensymbol, wobei Konstantensymbole Codierungen von Elementen aus $\mathcal{P}_{\text{fin}}(\mathbb{N})$ seien. Dann ist $X$ ein Term. Seien ferner $\beta, \gamma$ Terme, $O \subseteq \{+, \times, \cup, \cap, -\}$ und $\oplus \in O$. Dann ist $(\beta \oplus \gamma)$ ein Term. Sei $T$ die Menge aller dieser Terme.

Es seien $X, Y \in T$. Dann heißt $X = Y$ **Atom**.

Es seien $a_1 = (t_{1,1} = t_{1,2}), \ldots, a_m = (t_{m,1} = t_{m,2})$ für $m \in \mathbb{N}$ Atome. Ferner seien $X_1, \ldots, X_n$ die in den genannten Atomen auftretenden Variablensymbole. Dann nennen wir

$$\varphi = \exists X_1 \ldots \exists X_n \; a_1 \wedge \cdots \wedge a_m$$

$O$-**Formel** oder $O$-**Satz**. Ist $O$ aus dem Kontext ersichtlich, schreiben wir auch einfach Formel oder Satz. Die in $\varphi$ auftretenden Variablensymbole bezeichnen wir mit $\text{Var}_\varphi$ und die in $\varphi$ auftretenden Konstantensymbole mit $\text{Const}_\varphi$.
Des Weiteren sei mit $T_\varphi$ die Menge aller in $\varphi$ auftretenden Terme bezeichnet.

Wir definieren die Semantik von Termen. Eine totale Funktion $\alpha : T_\varphi \to \mathcal{P}_{\text{fin}}(\mathbb{N})$ heißt **Termbelegung** für $\varphi$, wenn die folgenden Bedingungen erfüllt sind.

- für Konstantensymbole $C$ ist $\alpha(C)$ das Element aus $\mathcal{P}_{\text{fin}}(\mathbb{N})$, welches durch $C$ codiert ist. Wir werden der Einfachheit halber $C$ mit $\alpha(C)$ identifizieren und $C$ somit auch als eine endliche Teilmenge natürlicher Zahlen auffassen.
- für alle Variablensymbole $X$ gilt $\alpha(X) \in \mathcal{P}_{\text{fin}}(\mathbb{N})$.
- für alle Terme $X+Y$ gilt: $\alpha(X+Y) = \alpha(X)+\alpha(Y) =_{\text{def}} \{x+y \mid x \in \alpha(X), y \in \alpha(Y)\}$.
- für alle Terme $X \times Y$ gilt: $\alpha(X \times Y) = \alpha(X) \times \alpha(Y) =_{\text{def}} \{x \cdot y \mid x \in \alpha(X), y \in \alpha(Y)\}$.
- für alle Terme $X \cup Y$ gilt: $\alpha(X \cup Y) = \alpha(X) \cup \alpha(Y)$.
- für alle Terme $X \cap Y$ gilt: $\alpha(X \cap Y) = \alpha(X) \cap \alpha(Y)$.
- für alle Terme $X - Y$ gilt: $\alpha(X - Y) = \alpha(X) - \alpha(Y) =_{\text{def}} \{x \mid x \in \alpha(X), x \notin \alpha(Y)\}$.

$\varphi$ ist genau dann **wahr**, wenn es eine Termbelegung $\alpha$ mit $\alpha(t_{i,1}) = \alpha(t_{i,2})$ für alle $i = 1, \ldots, m$ gibt. Dann nennen wir $\alpha$ **erfüllend** oder **gültig**.
Die Einschränkung einer Termbelegung $\beta$ auf die in $\varphi$ auftretetenden Variablensymbole $\text{Var}_\varphi$ heißt **Variablenbelegung** oder auch kurz **Belegung**. Es heißt $\beta$ **erfüllend** oder **gültig**, wenn die von $\beta$ induzierte Termbelegung erfüllend ist[1].

Mit diesen Begriffen ist es uns möglich, die für diese Arbeit zentralen Probleme zu definieren. Dies geschieht in der folgenden Definition.

**Definition 2.2**
Es sei $O \subseteq \{+, \times, \cup, \cap, -\}$. Dann ist

$$\text{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \text{O}\Big) =_{\text{def}} \{\varphi \mid \varphi \text{ ist eine wahre } O\text{-Formel}\}.$$

Weiterhin sei

$$\begin{aligned}\text{CSP}\Big([\mathbb{N}], \{=\} \cup \text{O}\Big) =_{\text{def}} \{\langle\varphi\rangle \mid\ & \varphi \text{ ist eine } O\text{-Formel, deren Konstanten Intervalle sind}^2, \\ & \text{und es existiert eine gültige Termbelegung } \alpha \text{ mit} \\ & \alpha(\text{Var}_\varphi \cup \text{Const}_\varphi) \subseteq [\text{N}]\ \}\end{aligned}$$

definiert.

Direkt aus der Definition ergibt sich das folgende triviale Lemma.

**Lemma 2.3**
*Für $O \subseteq O' \subseteq \{+, \times, \cup, \cap, -\}$ und $M \in \{\mathcal{P}_{\text{fin}}(\mathbb{N}), [N]\}$ gilt*

$$\text{CSP}\Big(\text{M}, \{=\} \cup \text{O}\Big) \leq_{\text{m}}^{\log} \text{CSP}\Big(\text{M}, \{=\} \cup \text{O}'\Big).$$

Wir betrachten im Folgenden ein weiteres Problem, das wir als Hilfsmittel verwenden, um zahlreiche Beweise zu vereinfachen.

[1] Man sieht leicht, dass es zu einer gegebenen Belegung der Variablen genau eine Termbelegung gibt, die auf der Menge der Variablen mit dieser übereinstimmt.

[2] Wir erinnern daran, dass die in diesem Kontext auftretenden Konstanten – da sie Intervalle sind – auch als solche codiert werden. Das heißt, dass eine Konstante $[a, b]$ für $a, b \in \mathbb{N}$ und $a \leq b$ so codiert ist, dass die Länge der Codierung der Konstante in $\Theta(|a| + |b|)$ ist.

**Definition 2.4**
Es sei $O \subseteq \{+, \times, \cup, \cap, -\}$. Wir definieren

$$\mathrm{CSP}'\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup O\Big) =_{\mathrm{def}} \{\varphi \in \mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big) \mid \text{jedes Atom ist von der Form } X \oplus Y = Z \text{ oder } Z = X \oplus Y^{3} \text{ für } X, Y, Z \in \mathrm{Const}_{\varphi} \cup \mathrm{Var}_{\varphi} \text{ und } \oplus \in O\}$$

Es sei $\mathrm{CSP}'\Big([\mathbb{N}], \{=\} \cup \mathrm{O}\Big)$ für $O \subseteq \{\cap, +\}$ analog wie $\mathrm{CSP}'\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big)$ definiert.

Das folgende Lemma erlaubt es uns, in vielen Situationen davon auszugehen, dass Eingabeformeln die geschilderte einfache Form besitzen. Es ist nämlich in einigen Fällen möglich, große Terme in mehrere kleine Terme zu zerlegen, indem man „Zwischenergebnisse“ in neu hinzugefügten Variablen speichert. Diese Vorgehensweise funktioniert bei CSPs über $[\mathbb{N}]$ jedoch offenbar nur dann, wenn die Menge $[\mathbb{N}]$ bezüglich der zugelassenen Operationen abgeschlossen ist.

**Lemma 2.5**
*Es sei $O \subseteq \{+, \times, \cup, \cap, -\}$. Dann gilt*

$$\mathrm{CSP}'\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big) \equiv_{m}^{\log} \mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big).$$

*Falls $O \subseteq \{+, \cap\}$, so gilt auch*

$$\mathrm{CSP}'\Big([\mathbb{N}], \{=\} \cup \mathrm{O}\Big) \equiv_{m}^{\log} \mathrm{CSP}\Big([\mathbb{N}], \{=\} \cup \mathrm{O}\Big).$$

*Beweis.* Wir zeigen zunächst den ersten Teil der Aussage. $\leq_{\mathrm{m}}^{\log}$ ist trivial. Wir zeigen die umgekehrte Reduktion.

Sei $\varphi = \exists Z_1 \ldots \exists Z_m \bigwedge_{i=1}^{n} \big(t_{i,1} = t_{i,2}\big)$ eine $O$-Formel. Wir skizzieren die Berechnung einer $O$-Formel $\varphi'$ mit

$$\varphi \in \mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big) \Leftrightarrow \varphi' \in \mathrm{CSP}'\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big).$$

- Sei $m_{i,j} \in \mathbb{N}$ die Auftretenshäufigkeit von Symbolen aus $O$ in $t_{i,j}$. Führe für jeden Term $t_{i,j}$ neue Variablen $X_{i,j,1}, \ldots, X_{i,j,m_{i,j}}$ ein. Quantifiziere am Anfang der Ausgabeformel über alle Variablen $X_{i,j,k_{i,j}}, Z_r$ für $i = 1, \ldots, n$, $j = 1, 2$, $k_{i,j} = 1, \ldots, m_{i,j}$ und $r = 1, \ldots, m$ existentiell.
- Für jeden Term $t_{i,j}$ und für $k = 1, \ldots, m_{i,j}$ führe das Folgende aus:
  - Es sei $\oplus_k$ das $k$-te Symbol (von links gezählt) aus $O$ im Term $t_{i,j}$.
  - Falls links von $\oplus_k$ keine Klammer, sondern direkt eine Variable oder Konstante steht, so bezeichne $L$ diese Variable oder Konstante.
  - Ist dies nicht der Fall, befindet sich links mindestens eine schließende Klammer direkt neben $\oplus_k$. Finde zu der schließenden Klammern direkt links von $\oplus_k$ die zugehörige öffnende Klammer. Zwischen diesen beiden Klammern befindet sich ein Term $\tau \oplus_s \tau'$ für zwei Terme $\tau, \tau'$ und $s \in \{1, \ldots, m_{i,j}\}$. In diesem Fall bezeichne $L$ die Variable $X_{i,j,s}$.

[3] Wenn wir im Folgenden über alle Atome $X \oplus Y = Z$ quantifizieren, sind damit auch alle Atome der Forme $Z = X \oplus Y$ gemeint. Wir werden dies ab jetzt nicht mehr explizit erwähnen.

  - $L$ bezeichnet somit den Term, welcher der linke Operand von $\oplus_k$ ist. Verfahre analog auf der rechten Seite, sodass $R$ den Term bezeichnet, welcher der rechte Operand von $\oplus_k$ ist.
  - Füge an die Ausgabeformel das Atom $L \oplus_k R = X_{i,j,k}$ an.

- Für jedes Atom $t_{i,1} = t_{i,2}$ seien $Y_1, Y_2$ die Variablen, die für $t_{i,1}$ bzw. $t_{i,2}$ stehen. Setze je nach Verfügbarkeit von Operationen $T_1 + \{0\} = T_2$, $T_1 \times \{1\} = T_2$, $T_1 \cup T_1 = T_2$, $T_1 \cap T_1 = T_2$ oder $T_1 - \emptyset = T_2$.

Es kann der Algorithmus in logarithmischem Raum berechnet werden, da zu jedem Zeitpunkt lediglich eine konstante Anzahl von Variablen oder Konstanten gespeichert werden muss und ansonsten nur die Anzahlen von Operationssymbolen in Termen sowie die Anzahlen von Klammern gezählt werden müssen.

Weiterhin hat in der Ausgabeformel jedes Atom die Form $X \oplus Y = Z$ für $\oplus \in O$ und Variablen- oder Konstantensymbole $X, Y, Z$.

Ist $\varphi$ wahr, so ist auch $\varphi'$ wahr: Sei $\alpha$ eine erfüllende Termbelegung für $\varphi$. Belege die Variablen $Z_1, \ldots, Z_m$ in $\varphi'$ mit $\alpha(Z_1), \ldots, \alpha(Z_m)$ und setze die so konstruierte Abbildung so fort, dass sie eine Termbelegung $\beta$ ist. Dann ist auch $\beta$ erfüllend.

Ist umgekehrt $\varphi'$ wahr, so sei $\beta$ eine erfüllende Termbelegung für $\varphi'$. Setze nun

$$\alpha : \mathrm{Var}_\varphi \to \mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \quad \mathrm{X} \mapsto \beta(\mathrm{X}).$$

Dann ist $\alpha$ eine erfüllende Variablenbelegung für $\varphi$ und es ist $\varphi$ wahr.

Der zweite Teil der Aussage kann analog gezeigt werden. Gibt es nämlich für eine $O$-Formel $\varphi \in \mathrm{CSP}\Big([\mathbb{N}], \{=\} \cup \mathrm{O}\Big)$ mit $O \subseteq \{+, \cap\}$ eine erfüllende Termbelegung $\alpha$, dann gilt wegen

$$I + J \in [\mathbb{N}] \quad \text{und} \quad I \cap J \in [\mathbb{N}] \quad \text{für } I, J \in [\mathbb{N}]$$

für jeden Term $t$

$$\alpha(t) \in [\mathbb{N}].$$

□

Wegen der Transitivität von $\leq_{\mathrm{m}}^{\log}$ dürfen wir im Folgenden bei logspace-Berechnungen und bei Berechnungen mit größerer Komplexität für eine $O$-Formel stets davon ausgehen, dass diese die Form aus Definition 2.4 hat.

Das folgende Resultat zeigt, dass es für CSPs über der Grundmenge $\mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ möglich ist, durch „$-$“ sowohl „$\cup$“ als auch „$\cap$“ zu simulieren.

**Lemma 2.6**
*Es gilt*

$$\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big) \leq_{\mathrm{m}}^{\log} \mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, -\} \cup (\mathrm{O} - \{\cup, \cap\})\Big)$$

*für* $O \subseteq \{+, \times, \cup, \cap\}$.

*Beweis.* Die Aussage gilt wegen Lemma 2.5 und weil $X = Y \cup Z$ auch durch

$$\big((X - Y) - Z = \emptyset\big) \wedge \big(Y - X = \emptyset\big) \wedge \big(Z - X = \emptyset\big)$$

sowie $X = Y \cap Z$ auch durch

$$\big(X - Y = \emptyset\big) \wedge \big(X - Z = \emptyset\big) \wedge \big((Y - (Y - Z)) - X = \emptyset\big)$$

ausgedrückt werden kann. □

Es genügt also die Probleme $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big)$ für $O \subseteq \{\cup, \cap, -, +, \times\}$ zu betrachten, für welche

$$- \in O \Rightarrow \{\cup, \cap\} \subseteq O$$

gilt.

# 3 CSPs mit ausschließlich Mengenoperationen

Wir beschränken uns zunächst auf CSPs, bei denen keine arithmetischen, sondern nur Mengenoperationen zugelassen sind. Diese Beschränkung macht die Situation in zweierlei Hinsicht wesentlich einfacher:

Zum einen wird sich herausstellen, dass die betrachteten Probleme alle innerhalb von NP liegen, während – wie wir in Abschnitt 4 sehen werden – CSPs NP-hart sind, sobald ein arithmetischer Operator zugelassen ist. Dabei werden wir auch die PSPACE- und die $\Sigma_1$-Härte einiger Probleme zeigen. Die in diesem Abschnitt untersuchten Probleme fallen also in kleinere Komplexitätsklassen.

Zum anderen gestaltet sich die Zuordnung zu einzelnen Komplexitätsklassen in diesem Abschnitt wesentlich leichter. Für jede mögliche Teilmenge $O \subseteq \{\cup, \cap, -\}$ können wir $\mathrm{CSP}\big(\mathrm{M}, \{=\} \cup \mathrm{O}\big)$ für $M \in \{\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), [\mathbb{N}]\}$ als $\leq_{\mathrm{m}}^{\log}$-vollständig für eine der Klassen L, P oder NP beweisen. Damit sind die genannten Probleme erschöpfend behandelt. Dies wird uns in Abschnitt 4 nicht möglich sein.

## 3.1 Zwei Spezialfälle

Wir beginnen mit der Betrachtung zweier Spezialfälle. Zunächst werden wir eine obere Schranke für den Fall zeigen, dass alle Mengenoperationen erlaubt sind. Dies wird gleichzeitig eine obere Schranke für alle in diesem Kapitel betrachteten Probleme darstellen. Anschließend werden wir CSPs untersuchen, in denen gar keine Operationen zugelassen sind.

### 3.1.1 Eine obere Schranke

Das folgende Lemma zeigt, dass alle CSPs, in denen keine arithmetischen Operationen auftreten, in NP sind.

**Lemma 3.1**
*Es sei* $O = \{-, \cup, \cap\}$ *und* $M \in \{[\mathbb{N}], \mathcal{P}_{\mathrm{fin}}(\mathbb{N})\}$. *Dann gilt* $\mathrm{CSP}\big(\mathrm{M}, \{=\} \cup \mathrm{O}\big) \in \mathrm{NP}$.

*Beweis.* Wir zeigen die Aussagen für $M = \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ und $M = [\mathbb{N}]$ parallel.
Es sei $\varphi \in \mathrm{CSP}\big(\mathrm{M}, \{=\} \cup \mathrm{O}\big)$. Weiterhin sei $K = \bigcup_{C \in \mathrm{Const}_\varphi} C$ im Fall $M = \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ und $K = [0, \max\big(\{x \in \bigcup_{C \in \mathrm{Const}_\varphi} C\}\big)]$ im Fall $M = [\mathbb{N}]$.
Sei $\alpha$ eine erfüllende Termbelegung für $\varphi$. Dann ist ebenfalls $\beta$ mit $\beta(X) = \alpha(X) \cap K$ eine erfüllende Termbelegung:

Mittels einer strukturellen Induktion über die Definition einer Termbelegung lässt sich leicht zeigen, dass $\beta$ eine Termbelegung ist.

Angenommen, $\beta$ wäre nicht erfüllend. Dann gäbe es ein Atom $t_1 = t_2$ mit Termen $t_1, t_2$, sodass $\alpha(t_1) = \alpha(t_2)$, aber $\alpha(t_1) \cap K \neq \alpha(t_2) \cap K$. O.B.d.A. existiere ein

$$x \in \big(\alpha(t_1) \cap K\big) - \big(\alpha(t_2) \cap K\big).$$

Also gilt $x \in K$ und somit $x \in \alpha(t_1) - \alpha(t_2)$, im Widerspruch zu $\alpha(t_1) = \alpha(t_2)$.

Wann immer also eine erfüllende Termbelegung für $\varphi$ existiert, gibt es auch eine erfüllende Termbelegung, in deren Wertebereich nur Mengen aus „kleinen“ Zahlen vorkommen, d.h. die auftretenden Zahlen sind kleiner oder gleich der größten in einer Konstanten enthaltenen Zahl. Daraus folgt unmittelbar, dass der folgende nichtdeterministische Polynomialzeitalgorithmus entscheidet, ob $\varphi \in \mathrm{CSP}\big(\mathrm{M}, \{=\} \cup \mathrm{O}\big)$.

1. Rate nichtdeterministisch eine Variablenbelegung mit Werten aus $\{A \in M \mid A \subseteq K\}$.
2. Teste, ob die Belegung erfüllend ist und gib entsprechend 0 oder 1 zurück.

Der Schritt 1 kann in Polynomialzeit ausgeführt werden, weil $K$ im Fall $M = \mathcal{P}_{\text{fin}}(\mathbb{N})$ lediglich $O(|\varphi|)$ Elemente enthält und im Fall $M = [\mathbb{N}]$ die Intervallgrenzen aus einer Menge mit lediglich polynomiell langen Zahlen ausgewählt werden müssen. □

**Bemerkung 3.2**
*Bei der Untersuchung von CSPs mit ausschließlich Mengenoperationen treten einige* NP*-vollständige Probleme auf. Durch das obige Lemma zusammen mit Lemma 2.3 haben wir bereits für jedes dieser Probleme die Zugehörigkeit zu* NP *gezeigt. Es genügt nun also, für jedes dieser Probleme die* $\leq_{\text{m}}^{\log}$*-Härte für* NP *zu zeigen. Die* $\leq_{\text{m}}^{\log}$*-Vollständigkeit folgt dann sofort.*

### 3.1.2 CSPs ohne Operationen

Nun betrachten wir den Fall, dass gar keine Operationen zur Verfügung stehen. Es ist $\text{CSP}\big(\text{M}, \{=\}\big)$ für $M \in \{\mathcal{P}_{\text{fin}}(\mathbb{N}), [\mathbb{N}]\}$[4] das einzige in dieser Arbeit betrachtete CSP, welches wir nicht als $\leq_{\text{m}}^{\log}$-hart für P beweisen können.
Wir zeigen für dieses Problem die Zugehörigkeit zur Komplexitätsklasse L. Dazu verwenden wir das Erreichbarkeitsproblem in ungerichteten Graphen.

**Definition 3.3**
Es sei $(V, E)$ ein ungerichteter Graph. Für $u, v \in V$ heißt $v$ **erreichbar** von $u$ aus, wenn $u = v$ gilt oder es $k \in \mathbb{N}^+$ und Knoten $v_1, v_2, \ldots, v_k$ mit $(u, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k), (v_k, v) \in E$ gibt.

Damit definieren wir

$$\text{USTCON} = \{\langle((V, E), u, v)\rangle \mid (V, E) \text{ ist ein ungerichteter Graph mit } u, v \in V \text{ und } u \text{ ist von } v \text{ aus erreichbar}\}.$$

Es war lange Zeit unklar, ob USTCON $\in$ L gilt. Die gerichtete Variante des Problems ist als NL-vollständig bekannt. Da ungerichtete Graphen spezielle gerichtete Graphen sind, ist zum einen klar, dass USTCON $\in$ NL gilt, zum anderen bestand die berechtigte Hoffnung, dass sich die Erreichbarkeit für den Spezialfall eines gerichteten Graphs leichter entscheiden lässt.
Papadimitriou [Pap94] etwa weist darauf hin, dass das Problem einfacher sei als die gerichtete Variante, da es in randomisiertem logarithmischem Raum gelöst werden könne, auch wenn es ihm nicht möglich sei, die Zugehörigkeit zu L zu zeigen.

Schließlich konnte Reingold [Rei05] zeigen, dass USTCON sogar in deterministischem logarithmischem Raum entschieden werden kann.

Er weist auch darauf hin, dass es in logarithmischem Raum berechenbare Transformationen zwischen natürlichen Repräsentationen von Graphen gebe und somit das Resultat zum großen Teil unabhängig von der Repräsentation des Eingabegraphs sei. Explizit erwähnt er in diesem Zusammenhang die Darstellung eines Graphen als Adjazenzmatrix.

Daher werden wir in der im Folgenden dargestellten Berechnung von USTCON-Instanzen von Adjazenzmatrizen als Repräsentationsform der Graphen ausgehen.

[4] Für den verbleibenden Abschnitt gehen wir von $M \in \{\mathcal{P}_{\text{fin}}(\mathbb{N}), [\mathbb{N}]\}$ aus, ohne dies erneut zu erwähnen.

Unter Verwendung von polynomiell vielen Anfragen an USTCON kann die folgende Aussage bewiesen werden:

**Lemma 3.4**
*Es gilt* $\mathrm{CSP}\big(\mathrm{M}, \{=\}\big) \in \mathrm{L}$.

*Beweis.* Es sei $\varphi$ eine $\emptyset$-Formel, also

$$\varphi = \exists X_1 \ldots \exists X_n \left(Y_1 = Y_2\right) \wedge \cdots \wedge \left(Y_{2m-1} = Y_{2m}\right)$$

für $n, m \in \mathbb{N}$ und $Y_i \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$.

Wir betrachten den ungerichteten Graphen $G_\varphi = (V_\varphi, E_\varphi)$ mit $V_\varphi = \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$ und

$$E_\varphi = \{\{Y_{2i-1}, Y_{2i}\} \mid i = 1, \ldots, m \wedge Y_{2i-1} \neq Y_{2i}{}^5\}.$$

Die totale Funktion $f$ mit $\varphi \mapsto G_\varphi$ ist in FL, da für jedes Atom lediglich die entsprechende Spalte und Zeile in der Adjazenzmatrix des Graphen zu suchen ist. Dies erfordert lediglich logarithmischen Speicher, da zu jedem Zeitpunkt höchstens zwei Verweise auf Konstanten oder Variablen zu speichern sind[6].

Wir zeigen

$$\varphi \text{ ist nicht wahr } \Leftrightarrow \exists C_1, C_2 \in \mathrm{Const}_\varphi \subseteq \mathrm{V}_\varphi : \big(\mathrm{C}_1 \neq \mathrm{C}_2 \wedge C_1 \text{ ist von } C_2 \text{ aus erreichbar}\big)$$

„⇒“: Wir zeigen die Kontraposition. Es gebe also für je zwei verschiedene Konstanten $C_1, C_2$ keinen Pfad von $C_1$ nach $C_2$. Wir konstruieren iterativ eine erfüllende Termbelegung $\alpha$. Setze $\alpha(C) = C$ für jede Konstante $C$. Setze ferner iterativ für jede Variable $X$, für die ein $Y \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$ mit

- definiertem $\alpha(Y)$ und
- einem Atom $X = Y$ bzw. $Y = X$ in $\varphi$

existiert, $\alpha(X) = \alpha(Y)$. Setze dann, nachdem das geschilderte iterative Verfahren stationär geworden ist, für alle verbleibenden Variablen $X$

$$\alpha(X) = \emptyset.$$

Es wird also jede Variable mit $\emptyset$ oder mit dem Wert einer Konstanten belegt. Wäre nun ein Atom $Y_{2i-1} = Y_{2i}$ von $\varphi$ nicht erfüllt, so wäre nach der Konstruktion von $\alpha$

$$\alpha(Y_{2i-1}) \neq \emptyset \wedge \alpha(Y_{2i}) \neq \emptyset$$

und somit $\alpha(Y_{2i-1}), \alpha(Y_{2i}) \in \mathrm{Const}_\varphi$. Nach Konstruktion von $\alpha$ müsste es dann aber einen Pfad[7] von einer Konstanten $C$ mit $C = \alpha(Y_{2i-1})$ zu $Y_{2i-1}$ und weiterhin einen Pfad von einer Konstanten $C' = \alpha(Y_{2i})$ zu $Y_{2i}$ geben. Dann gäbe es aber auch einen Pfad von $C$ zu $C'$, was wegen $C = \alpha(Y_{2i-1}) \neq \alpha(Y_{2i}) = C'$ ein Widerspruch wäre.

[5] Wir gehen hierbei davon aus, dass Konstanten mit dem gleichen Wert auch gleich codiert sind.

[6] Die jeweilige Konstante kann zwar nicht notwendig in logarithmischem Raum gespeichert werden, es genügt jedoch ein Zeiger auf diese. Der Gleichheitstest für zwei Konstanten mit -sagen wir- linearer Länge ist offenbar in logarithmischem Raum möglich.

[7] Wir verwenden hier den Begriff Pfad so, dass es in jedem Graphen für jeden Knoten einen Pfad zu sich selbst gibt, nämlich den trivialen Pfad, der keine Kante verwendet.

„⇐“: Erreichbarkeit zweier Knoten $C_1, C_2 \in \text{Const}_\varphi \subseteq \text{V}$ in $G_\varphi$ bedeutet, dass für jede erfüllende Termbelegung $\alpha$

$$\alpha(C_1) = \alpha(C_2)$$

gilt. Wegen $C_1 \neq C_2$ gibt es daher keine erfüllende Termbelegung und $\varphi$ ist nicht wahr.

Betrachte den folgenden Algorithmus:

- Eingabe: $\emptyset$-Formel $\varphi$
- Für je zwei Konstanten $C, C'$:
  - Falls $C$ von $C'$ aus in $f(\varphi)$ erreichbar ist, gib 0 zurück.
- Gib 1 zurück.

Es ist hinreichend zu zeigen, dass der Schleifenrumpf in deterministisch logarithmischem Raum ausgeführt werden kann. Da mit $c_{\text{USTCON}}(f(\varphi), C, C')$ lediglich eine Verkettung von polynomiell vielen logspace-Berechnungen zu berechnen ist, gilt dies. Es folgt daher $\text{CSP}\big(\text{M}, \{=\}\big) \in \text{L}$. □

Mit Lemma 3.4 ist trivialerweise klar, dass $\text{CSP}\big(\text{M}, \{=\}\big)$ $\leq_{\text{m}}^{\log}$-vollständig für L ist. Dennoch zeigen wir die Reduktion $\overline{\text{USTCON}} \leq_{\text{m}}^{\log} \text{CSP}\big(\text{M}, \{=\}\big)$, um zu zeigen, dass ein direkter Beweis von $\text{CSP}\big(\text{M}, \{=\}\big) \in \text{L}$ genauso schwierig wie ein Beweis für $\text{USTCON} \in \text{L}$ ist.

Es sei $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ ein ungerichteter Graph und $v_i, v_j \in V$. Falls $i = j$, so gib $\emptyset = \{0\}$ zurück.
Andernfalls gib

$$\varphi = \exists V_1 \ldots \exists V_n \ \big(V_i = \{0\}\big) \wedge \big(V_j = \{1\}\big) \wedge \bigwedge_{(v_s, v_t) \in E} \big(V_s = V_t\big)$$

zurück.
Mit einer ähnlichen Argumentation wie im Beweis von Lemma 3.4 lässt sich zeigen, dass es genau dann einen Pfad von $v_i$ nach $v_j$ gibt, wenn die zurückgegebene Formel nicht wahr ist.

## 3.2 Vereinigung

In diesem Abschnitt zeigen wir die $\leq_{\mathrm{m}}^{\log}$-Vollständigkeit von $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup\}\Big)$ für P und die $\leq_{\mathrm{m}}^{\log}$-Vollständigkeit von $\mathrm{CSP}\Big([\mathbb{N}], \{=, \cup\}\Big)$ für NP.

### 3.2.1 $\mathrm{CSP}\Big(\mathcal{P}_{\mathbf{fin}}(\mathbb{N}), \{=, \cup\}\Big)$

**Lemma 3.5**
*Es gilt* $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup\}\Big) \in \mathrm{P}$.

*Beweis.* Nach Lemma 2.5 genügt es, für eine $\{\cup\}$-Formel $\varphi$ in Polynomialzeit zu entscheiden, ob $\varphi$ in $\mathrm{CSP}'\Big((\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup\}\Big)$ liegt. Es sei also

$$\varphi = \exists X_1 \ldots \exists X_k \bigwedge_{i=1}^{t} (A_i \cup B_i = C_i),$$

mit $k, t \in \mathbb{N}$ und Variablen- oder Konstantensymbolen $A_i, B_i, C_i$.

Der folgende Algorithmus entscheidet, ob es eine erfüllende Belegung der Variablen gibt. Dazu sei

$$K = \bigcup_{C \in \mathrm{Const}_\varphi} C.$$

1. Setze $\alpha(X) = \begin{cases} X & \text{falls } X \text{ Konstante} \\ K & \text{sonst} \end{cases}$
2. Wende für jedes Atom $X \cup Y = Z$ die folgenden Regeln an.
   (a) Setze $\alpha(Z) = \alpha(Z) \cap (\alpha(X) \cup \alpha(Y))$.
   (b) Setze $\alpha(X) = \alpha(X) \cap \alpha(Z)$ und analog $\alpha(Y) = \alpha(Y) \cap \alpha(Z)$.
   (c) Falls $\alpha(C)$ für eine Konstante $C$ geändert wurde, gib 0 zurück.
3. Hat sich in Schritt 2 eine Änderung eines Wertes $\alpha(X)$ für ein $X \in \mathrm{Var}_\varphi$ ergeben, so führe Schritt 2 erneut aus.
4. Gib 1 zurück.

Um zu sehen, dass der Algorithmus ein Polynomialzeitalgorithmus ist, genügt es, zu zeigen, dass der Schritt 2 nur polynomiell oft ausgeführt wird. Dies ergibt sich direkt daraus, dass es für jede Variable $X$ höchstens $|K|$ Änderungen des Wertes $\alpha(X)$ gibt.

Es bleibt zu zeigen, dass der Algorithmus genau dann 1 zurückgibt, wenn $\varphi$ wahr ist.

„⇒“: Es gebe der Algorithmus 1 zurück. Mit $\alpha(X)$ bezeichnen wir im Folgenden den Wert für eine Variable $X$ zum Ende der Ausführung des Algorithmus. Wir zeigen nun, dass die Belegung $\alpha$ so konstruiert wurde, dass für jedes Atom $A_i \cup B_i = C_i$ die Gleichung $\alpha(A_i) \cup \alpha(B_i) = \alpha(C_i)$ erfüllt ist.
Angenommen, es sind nicht alle Gleichungen $\alpha(A_i) \cup \alpha(B_i) = \alpha(C_i)$ erfüllt. Dann gibt es ein $j$, sodass $\alpha(A_j) \cup \alpha(B_j) \neq \alpha(C_j)$ gilt. Wir schreiben im Folgenden $A, B, C$ für $\alpha(A_j), \alpha(B_j), \alpha(C_j)$ und unterscheiden die folgenden Fälle.

- $\exists x \in C - (A \cup B)$:

  Dann hätte bei der letzten Ausführung von Schritt 2, bei der sich nichts mehr geändert hat, die Regel 2a
  $$C = C \cap \big(A \cup B\big)$$
  angewendet werden müssen und es wäre daher $x \notin C$.

- $\exists x \in \big(A \cup B\big) - C$:

  In dieser Situation hätten noch die Regeln aus 2b
  $$A = A \cap C \quad B = B \cap C$$
  angewendet werden müssen, sodass $x \notin A \cup B$ gegolten hätte.

Wir erhalten somit jeweils Widersprüche. Folglich ist $\varphi$ wahr.

„$\Leftarrow$“: Es sei $\beta : \mathrm{Const}_\varphi \cup \mathrm{Var}_\varphi \to \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ eine erfüllende Belegung, wobei $\beta(C) = C$ für alle $C \in \mathrm{Const}_\varphi$ gelte. Wir können davon ausgehen, dass $\beta(\mathrm{Const}_\varphi \cup \mathrm{Var}_\varphi) \subseteq \mathcal{P}(\mathrm{K})$ gilt, da andernfalls $\beta(X)$ durch $\beta(X) \cap K$ ersetzt werden könnte und alle Gleichungen so erfüllt blieben.
Vor der ersten Ausführung von Schritt 2 gilt offenbar

$$\forall X \in \mathrm{Const}_\varphi \cup \mathrm{Var}_\varphi : \beta(\mathrm{X}) \subseteq \alpha(\mathrm{X}). \qquad (*)$$

Wir zeigen: Gilt $(*)$ vor der Anwendung einer der Regeln, dann auch danach.

- Regel 2a: Wir bezeichnen mit $U_v$ den Wert von $\alpha(Z)$ vor Anwendung der Regel und mit $U_n$ den Wert nach der Anwendung. $\alpha(X)$ und $\alpha(Y)$ bezeichnen die jeweiligen Wert zum Zeitpunkt der Anwendung der Regel, die Werte ändern sich durch Anwendung der Regel nicht. Falls $U_v = U_n$, so gilt $(*)$ offenbar auch nach Anwendung der Regel. Andernfalls existiert ein $x \in U_v - U_n$, also gilt $x \notin \alpha(X) \cup \alpha(Y)$. Dann folgt aber wegen $(*)$
  $$x \notin \beta(X) \cup \beta(Y),$$
  also auch $x \notin \beta(Z)$, weswegen $(*)$ auch nach Anwendung der Regel gilt.

- Regel 2b: Offenbar genügt es hier, nur die erste der beiden Gleichungen zu betrachten. Wir bezeichnen mit $U_v$ den Wert von $\alpha(X)$ vor der Anwendung der Regel und mit $U_n$ den Wert nach der Anwendung. $\alpha(Z)$ bezieht sich auf den Wert zum Zeitpunkt der Anwendung der Regel, dieser ändert sich durch die Anwendung der Regel nicht. Falls $U_v = U_n$, so gilt $(*)$ auch nach Anwendung der Regel. Andernfalls existiert ein $x \in U_v - U_n$, also gilt $x \notin \alpha(Z)$. Dann folgt aber wegen $(*)$
  $$x \notin \beta(Z),$$
  also auch $x \notin \beta(X)$, weswegen $(*)$ auch nach Anwendung der Regel gilt.

Insbesondere gilt also für jede Konstante $C$ zu jedem Zeitpunkt der Ausführung des Algorithmus $C = \beta(C) \subseteq \alpha(C) \subseteq C$, also wird der Wert von $\alpha(C)$ nicht verändert und der Algorithmus gibt 1 zurück. □

Es ist $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup\}\Big)$ sogar $\leq_{\mathrm{m}}^{\log}$-hart für P. Dies zeigt das folgende Lemma.

**Lemma 3.6**
$\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup\}\Big)$ *ist* $\leq_{\mathrm{m}}^{\log}$*-hart für* P.

*Beweis.* Wir verwenden das aus als P-vollständig bekannte Problem Monotone Circuit Value Evaluation[GHR91], welches wir mit MCVE bezeichnen. Wir werden $\mathrm{MCVE} \leq_{\mathrm{m}}^{\log} \mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup\}\Big)$ zeigen.
Wir betrachten den folgenden Algorithmus:

- Eingabe: Codierung $\overline{\alpha}$ eines monotonen booleschen Schaltkreises $\alpha$ mit Eingaben $x_1, \ldots, x_n \in \{0, 1\}$. Es seien die or-Gatter mit $o_1, \ldots, o_k$ und die and-Gatter mit $a_1, \ldots, a_m$ bezeichnet, wobei o.B.d.A. $o_k$ das Ausgabe-Gatter sei.
- Wir beschreiben sukzessive eine Formel $\varphi$. Es sei

$$\begin{aligned}\varphi =& \exists X_1 \ldots \exists X_n \exists O_1 \ldots \exists O_k \exists A_1 \exists H_1 \exists H_1' \ldots \exists A_m \exists H_m \exists H_m' \\ & \psi(X_1, \ldots, X_n, O_1, \ldots, O_k, A_1, \ldots, A_m, H_1, H_1', \ldots, H_m, H_m').\end{aligned}$$

  Mit jedem Eingang und jedem or- und and-Gatter korrespondiert also eine Variable, wobei es zu jedem and-Gatter zusätzlich noch zwei Hilfsvariablen gibt.
- Setze $\psi(X_1, \ldots, X_n, O_1, \ldots, O_k, A_1, \ldots, A_m, H_1, H_1' \ldots, H_m, H_m')$ auf

$$\bigwedge_{i=1}^{n} \Big(X_i = \begin{cases}\{1\} & \text{falls } x_i = 1 \\ \emptyset & \text{sonst}\end{cases}\Big) \wedge \bigwedge_{i=1}^{k} \chi_{o_i} \wedge \bigwedge_{i=1}^{m} \chi_{a_i} \wedge (O_k = \{1\}).$$

  Es steht dabei im Folgenden die Menge $\{1\}$ für die 1 im booleschen Schaltkreis und die Menge $\emptyset$ für eine 0.
- Ist $o_i$ ein or-Gatter und sind $A, B$ die Variablen, die für die Eingänge von $o_i$ stehen, so setze $\chi_{o_i} = (O_i = A \cup B)$.
- Ist $a_i$ ein and-Gatter und sind $A, B$ die Variablen, die für die Eingänge von $a_i$ stehen, so setze $\chi_{a_i} = (A_i \cup H_i = A) \wedge (A_i \cup H_i' = B)$.
- Gib $f(\overline{\alpha}) = \varphi$ zurück.

Wir bezeichnen die vom Algorithmus berechnete Funktion mit $f$.

Es kann $f$ in logarithmischem Raum berechnet werden, da stets nur konstant viele Variablen oder Konstanten gemerkt werden müssen. Ferner ist leicht ersichtlich, dass für einen Schaltkreis $\alpha$, der bei Eingabe $x_1, \ldots, x_n$ den Wert 1 ausgibt, die Formel $f(\overline{\alpha})$ wahr ist: Es müssen lediglich alle Variablen $O_i$ und $A_j$ mit dem Wert belegt werden, der für den Ausgabewert der entsprechenden Gatter $o_i$ und $a_j$ steht, also $\{1\}$ für 1 und $\emptyset$ für 0. Für or-Gatter erfordert die Formel $\varphi$ dies, für and-Gatter ist es möglich, eine Belegung zu wählen, die alle Atome der Formel erfüllt. Es bleibt somit die Rückrichtung zu zeigen:

Sei also $\varphi = f(\overline{\alpha})$ für die Codierung $\overline{\alpha}$ eines monotonen Schaltkreises $\alpha$ mit Eingabe $x_1, \ldots, x_n$ eine wahre Formel. Wir zeigen, dass der Schaltkreis $\alpha$ bei Eingabe $x_1, \ldots, x_n$ den Wert 1 ausgibt.
Da $\varphi$ wahr ist, existiert ein totales

$$\beta : \{X_1, \ldots, X_n, O_1, \ldots, O_k, A_1, \ldots, A_m, H_1, \ldots, H_m, H_1', \ldots, H_m'\} \to \mathcal{P}_{\mathrm{fin}}(\mathbb{N}),$$

sodass

$$\psi(\beta(X_1),\ldots,\beta(X_n),\beta(O_1),\ldots,\beta(O_k),\beta(A_1),\ldots,\beta(A_m),\beta(H_1),\beta(H_1'),\ldots,\beta(H_m),\beta(H_m'))$$

wahr ist. Dann gilt $W_\beta \subseteq \{\{1\},\emptyset\}$:
Dies hat seine Ursache darin, dass die $\beta(X_i)$ laut Formel einen dieser Werte annehmen müssen und alle $\beta(O_i)$, $\beta(A_j)$, $\beta(H_j)$, $\beta(H_j')$ eine Teilmenge der Vereinigung der beiden „eingehenden Variablen“ sein müssen.

Sei $\gamma : \{o_1,\ldots,o_k,a_1,\ldots,a_m\} \to \{0,1\}$ die Funktion, die jedem Gatter des Schaltkreises $\alpha$ bei Eingabe $x_1,\ldots,x_n$ seinen Ausgabewert zuordnet.
Wir zeigen induktiv die folgenden Aussagen, wobei $\max(\emptyset) = 0$ sei

(A) $\max(\beta(X_r)) \le x_r, \qquad r = 1,\ldots,n,$
(B) $\max(\beta(O_i)) \le \gamma(o_i), \qquad i = 1,\ldots k$ und
(C) $\max(\beta(A_i)) \le \gamma(a_i), \qquad j = 1,\ldots,m.$

Die Aussagen in (A) sind klar. Es sei nun ein Gatter $g$ mit der korrespondierenden Variable $G$ gegeben. Die beiden Eingänge seien Eingänge des Schaltkreises, or- oder and-Gatter und es gelte für ihre Werte $a, b$ und die zu ihnen korrespondierenden Variablen $A, B$

$$\max(\beta(A)) \le a \quad \text{und} \quad \max(\beta(B)) \le b.$$

Wir unterscheiden zwei Fälle:

- $g$ ist ein or-Gatter. Falls $\gamma(g) = 1$, gilt die Bedingung trivialerweise. Andernfalls gilt $a = b = 0$, also $\beta(A) = \emptyset$ und $\beta(B) = \emptyset$ und somit auch $\beta(G) = \beta(A) \cup \beta(B) = \emptyset$.
- $g$ ist ein and-Gatter. Falls $\gamma(g) = 1$ gilt die Bedingung trivialerweise. Andernfalls gilt o.B.d.A. $a = 0$ und somit $\beta(A) = \emptyset$. Wegen $\beta(G) \subseteq \beta(A)$ gilt $\beta(G) = \emptyset$.

Aus $1 = \max(\beta(O_k)) \le \gamma(o_k) \le 1$ folgt $\gamma(o_k) = 1$ und damit die Behauptung. □

Nun lässt sich der folgende Satz formulieren.

**Satz 3.7**
*Es ist* $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=,\cup\}\Big) \le_{\mathrm{m}}^{\log}$*-vollständig für* P.

*Beweis.* Die Aussage folgt aus den Lemmata 3.5 und 3.6. □

### 3.2.2 $\mathbf{CSP}\Big([\mathbb{N}], \{=,\cup\}\Big)$

Wider Erwarten ist für $O = \{\cup\}$ das Problem $\mathrm{CSP}([\mathbb{N}], \{=\} \cup \mathrm{O})$ schwerer als das Problem $\mathrm{CSP}([\mathcal{P}_{\mathrm{fin}}(\mathbb{N})], \{=\} \cup \mathrm{O})$[8]. Weil die Variante über $[\mathbb{N}]$ zuerst wie eine Einschränkung der Variante über $\mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ erscheint, würde man auf den ersten Blick eher das Gegenteil erwarten. Die $\{\cup\}$-Formeln über $[\mathbb{N}]$ sind auch tatsächlich eine eingeschränkte Version der Formeln über $\mathcal{P}_{\mathrm{fin}}(\mathbb{N})$, da der einzige Unterschied darin besteht, dass die Konstanten eine bestimmte Form aufweisen müssen.
Dadurch, dass in der Variante über den Intervallen jedoch auch die Variablen nur mit Intervallen belegt werden dürfen, erhalten wir in dieser speziellen Situation eine größere Ausdruckskraft.

[8] Diese Aussage gilt zumindest dann, wenn P ≠ NP gilt.

So können wir etwa mit dem Satz $\exists X \exists Y\, [0] \cup [2] = X \cup Y$ ausdrücken, dass genau eine der zwei Variablen unter einer erfüllenden Variablenbelegung den Wert [0] annimmt, während die andere den Wert [2] zugewiesen bekommt.
Dies ist nicht möglich, sobald Variablen beliebige endliche Teilmengen zugewiesen werden können.

Mit Hilfe derartiger Tricks können wir $3\text{SAT} \leq_{\text{m}}^{\log} \text{CSP}\Big([\mathbb{N}], \{=, \cup\}\Big)$ zeigen.

**Lemma 3.8**
*Es gilt* $3\text{SAT} \leq_{\text{m}}^{\log} \text{CSP}\Big([\mathbb{N}], \{=, \cup\}\Big)$.

*Beweis.* Sei $H$ eine aussagenlogische Formel in konjunktiver Normalform mit genau drei Literalen pro Klausel und Variablen $x_1, \ldots, x_n$, also $H = \bigwedge_{i=1}^{m} K_i$ für Klauseln $K_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ mit Literalen $l_{i,j}$ aus der Menge $\{x_1, \neg x_1, x_2, \neg x_2, \ldots, x_n, \neg x_n\}$.

Wir skizzieren eine $\{\cup\}$-Formel $\varphi$ mit $\varphi \in \text{CSP}\Big([\mathbb{N}], \{=, \cup\}\Big) \Leftrightarrow \text{H} \in 3\text{SAT}$. In dieser wird zunächst über die Variablen $X_r$,$X'_r$,$L_{i,1}$,$L_{i,2}$, $L_{i,3}$ existenziell quantifiziert, wobei $r = 1, \ldots, n$ und $i = 1, \ldots, m$.
Es gibt also für jede aussagenlogische Variable zwei Variablen in $\varphi$. Diese Variablen dienen dazu, für die zugehörigen aussagenlogischen Variablen den jeweiligen Wahrheitswert sowie dessen Negation zu beschreiben. Dabei wird der Wert [2] für den Wahrheitswert 1 und der Wert [0] für den Wahrheitswert 0 stehen. Wir beschreiben die weitere Formel $\varphi$:

$$\bigwedge_{r=1}^{n} \Big( X_r \cup X'_r = [0] \cup [2] \Big)$$

Unter einer erfüllenden Variablenbelegung hat somit jede Variable von $\varphi$ einen Wert aus $\{[0], [2]\}$ und eine Variable $X_r$ hat genau dann den Wert [0], wenn $X'_r$ den Wert [2] hat. Es beschreibt also $X_r$ den Wert von $x_r$ und $X'_r$ den Wert von $\neg x'_r$.
Es ordne $s : \{(i,j) \mid i \in \{1, \ldots, m\}, j \in \{1,2,3\}\} \to \{1, \ldots, n\}$ dem Paar $(i,j)$ den Index der Variable in $l_{i,j}$ zu. Nun ist sicherzustellen, dass jedes $L_{i,j}$ den richtigen Wert aus $\{X_{s(i,j)}, X'_{s(i,j)}\}$ annimmt. Dies fordern wir in $\varphi$ durch

$$\bigwedge_{i=1}^{m} \bigwedge_{j=1}^{3} \Big( L_{i,j} = \begin{cases} X_{s(i,j)} & \text{falls } l_{i,j} = x_{i,j} \\ X'_{s(i,j)} & \text{falls } l_{i,j} = \neg x_{i,j} \end{cases} \Big).$$

Wenn die aussagenlogischen Variablen von $\psi$ so mit 0 oder 1 belegt werden können, dass in jeder Klausel mindestens ein Literal den Wahrheitswert 1 hat, ist $\psi$ erfüllbar. Dies kann in $\varphi$ wie folgt beschrieben werden:

$$\bigwedge_{1 \leq i \leq m} \Big( L_{i,1} \cup L_{i,2} \cup L_{i,3} \cup [0] = [0] \cup [2] \Big).$$

Damit gilt

$$\varphi \in \text{CSP}\Big([\mathbb{N}], \{=, \cup\}\Big) \Leftrightarrow \text{H} \in \text{SAT}.$$

Mit der Feststellung, dass zu jedem Zeitpunkt der Ausführung des Algorithmus lediglich konstant viele Indizes der polynomiell vielen Variablen zu speichern sind und damit lediglich logarithmischer Raum für die Berechnung der geschilderten Reduktion benötigt wird, ist der Beweis vollständig.

□

Nun lässt sich leicht die NP-Vollständigkeit von $\mathrm{CSP}\Big([\mathbb{N}], \{=, \cup\}\Big)$ zeigen.

**Satz 3.9**
$\mathrm{CSP}\Big([\mathbb{N}], \{=, \cup\}\Big)$ *ist* $\leq_{\mathrm{m}}^{\log}$*-vollständig für* NP.

*Beweis.* Die Härte ergibt sich aus Lemma 3.8. Die Zugehörigkeit zu NP ergibt sich aus Bemerkung 3.2. □

## 3.3 Schnitt

Analog zu Abschnitt 3.2.1 kann wie für $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup\}\Big)$ auch für das Problem $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cap\}\Big)$ die $\leq_{\mathrm{m}}^{\log}$-Vollständigkeit für P gezeigt werden. Für die Variante über endlichen Intervallen können wir jedoch nicht wie für $\mathrm{CSP}\Big([\mathbb{N}], \{=, \cup\}\Big)$ die NP-Vollständigkeit zeigen. Wir werden vielmehr sehen, dass $\mathrm{CSP}\Big([\mathbb{N}], \{=, \cap\}\Big)$ vollständig für P ist.

### 3.3.1 $\mathrm{CSP}\Big(\mathcal{P}_{\mathbf{fin}}(\mathbb{N}), \{=, \cap\}\Big)$

**Lemma 3.10**
*Es gilt* $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cap\}\Big) \in \mathrm{P}$.

*Beweis.* Wie im Beweis von Lemma 3.5 genügt es gemäß Lemma 2.5, in Polynomialzeit zu entscheiden, ob $\varphi \in \mathrm{CSP}'\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cap\}\Big)$ für eine Formel

$$\varphi = \exists X_1 \ldots \exists X_k \bigwedge_{i=1}^{t} A_i \cap B_i = C_i,$$

mit $k, t \in \mathbb{N}$ und Variablen- oder Konstantensymbolen $A_i, B_i, C_i$ gilt.

Der folgende Algorithmus entscheidet, ob es ein totales $\alpha : \mathrm{Const}_\varphi \cup \mathrm{Var}_\varphi \to \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ mit $\alpha(c) = c$ für alle Konstanten gibt, sodass die Gleichungen $\alpha(A_i) \cup \alpha(B_i) = \alpha(C_i)$ gelten.

1. Setze $\alpha(X) = \begin{cases} X & \text{falls } X \text{ Konstante} \\ \emptyset & \text{sonst} \end{cases}$.
2. Wende für jede Gleichung $X \cap Y = Z$ die folgenden Regeln an.
   (a) Setze $\alpha(Z) = \alpha(Z) \cup (\alpha(X) \cap \alpha(Y))$.
   (b) Setze $\alpha(X) = \alpha(X) \cup \alpha(Z)$ und analog $\alpha(Y) = \alpha(Y) \cup \alpha(Z)$.
   (c) Falls $\alpha(C)$ für eine Konstante $C$ geändert wurde, gib 0 zurück.
3. Hat sich in Schritt 2 eine Änderung eines Wertes $\alpha(X)$ für ein $X \in \mathrm{Var}_\varphi$ ergeben, so führe Schritt 2 erneut aus.
4. Gib 1 zurück.

Wie im Beweis von Lemma 3.5 lässt sich nun zeigen, dass der Algorithmus genau dann 1 zurückgibt, wenn $\varphi$ wahr ist. □

Durch eine ähnliche Reduktion wie in Lemma 3.6 lässt sich die $\leq_{\mathrm{m}}^{\log}$-Härte des Problems $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cap\}\Big)$ für P zeigen.

**Lemma 3.11**
$\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cap\}\Big)$ *ist* $\leq_{\mathrm{m}}^{\log}$*-hart für* P.
$\mathrm{CSP}\Big([\mathbb{N}], \{=, \cap\}\Big)$ *ist* $\leq_{\mathrm{m}}^{\log}$*-hart für* P.

*Beweis.* Wir zeigen beide Behauptungen parallel. Der folgende Algorithmus berechnet eine Reduktionsfunktion von MCVE auf die beiden genannten Probleme.

- Eingabe: Codierung $\overline{\alpha}$ eines monotonen booleschen Schaltkreises $\alpha$ mit Eingaben $x_1, \ldots, x_n \in \{0,1\}$. Es seien die or-Gatter mit $o_1, \ldots, o_k$ und die and-Gatter mit $a_1, \ldots, a_m$ bezeichnet, wobei o.B.d.A. $o_k$ das Ausgabe-Gatter sei.
- Wir bauen sukzessive eine Formel $\varphi$ auf. Dazu sei

$$\begin{aligned}\varphi =& \exists X_1 \ldots \exists X_n \exists O_1 \ldots \exists O_k \exists A_1 \ldots \exists A_m \exists H_1 \exists H_1' \ldots \exists H_m \exists H_m' \\ & \psi(X_1, \ldots, X_n, O_1, \ldots, O_k, A_1, \ldots, A_m, H_1, H_1' \ldots, H_m, H_m').\end{aligned}$$

  Mit jedem Eingang und jedem or- und and-Gatter korrespondiert also eine Variable, wobei es zu jedem and-Gatter zusätzlich noch zwei Hilfsvariablen gibt.
- Setze $\psi(X_1, \ldots, X_n, O_1, \ldots, O_k, A_1, \ldots, A_m, H_1, H_1' \ldots, H_m, H_m')$ auf

$$\bigwedge_{i=1}^{n} \Big( X_i = \begin{cases} \{1\} & \text{falls } x_i = 0 \\ \emptyset & \text{sonst} \end{cases} \Big) \wedge \bigwedge_{i=1}^{k} \chi_{o_i} \wedge \bigwedge_{i=1}^{m} \chi_{a_i} \wedge (O_k = \emptyset).$$

- Ist $o_i$ ein or-Gatter und sind $A, B$ die Variablen, die für die Eingänge von $o_i$ stehen, so setze $\chi_{o_i} = (O_i = A \cap B)$.
- Ist $a_i$ ein and-Gatter und sind $A, B$ die Variablen, die für die Eingänge von $a_i$ stehen, so setze $\chi_{a_i} = \big(A_i \cap H_i = A\big) \wedge \big(A_i \cap H_i' = B\big)$.
- Gib $f(\overline{\alpha}) = \varphi$ zurück.

Wir bezeichnen die vom Algorithmus berechnete Funktion mit $f$. Es kann $f$ in logarithmischem Raum berechnet werden, da stets nur konstant viele Variablen- oder Konstantensymbole zu speichern sind.

Wir zeigen ferner

$$\overline{\alpha} \in \text{MCVE} \Leftrightarrow f(\overline{\alpha}) = \varphi \in \text{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, \cap\}\Big) \Leftrightarrow \text{f}(\overline{\alpha}) \in \text{CSP}\Big([\mathbb{N}], \{=, \cap\}\Big).$$

Die rechte Äquivalenz gilt, da in $\varphi$ lediglich Intervalle als Konstanten auftreten und zudem bei einer erfüllenden Belegung alle Variablen $X_i, A_i, O_i$ mit Teilmengen von [1] belegt werden müssen und daher im Falle der Existenz einer erfüllenden Belegung diese so gewählt werden kann, dass alle Variablen mit Teilmengen von [1] belegt sind.

Es bleibt die linke Äquivalenz zu zeigen:

„⇒“: Wenn $\overline{\alpha} \in$ MCVE gilt, also der Schaltkreis den Wert 1 ausgibt, dann können alle Variablen $A_i, O_i$ mit $\emptyset$ belegt werden, wenn das entsprechende Gatter den Wert 1 liefert, und andernfalls mit $\{1\}$. Die $H_i, H_i'$ können alle mit $\{1\}$ belegt werden. Dann gilt $\gamma(O_k) = \emptyset$ für die von uns beschriebene Belegung $\gamma$. Zudem sieht man leicht, dass für jedes „and“- und für jedes „or“-Gatter die zugehörige Bedingung $\chi_{a_i}$ bzw. $\chi_{o_i}$ in $\varphi$ erfüllt ist.
Es gilt also $\varphi \in \text{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, \cap\}\Big)$.

„⇐“: Sei nun $\overline{\alpha} \notin$ MCVE, der Schaltkreis berechne also im Gatter $o_k$ den Wert 0. Per Induktion über die Tiefe eines Gatters lässt sich für jedes Gatter $c$ mit zugehöriger Variable $C$ in $\varphi$ und eine beliebige Terminterpretation $\beta$

$$\beta(C) = \emptyset \Rightarrow c \text{ liefert den Wert 1}$$

zeigen. Weil $o_k$ den Wert 0 liefert, gilt für jede Terminterpretation $\beta$

$$\beta(O_k) \neq \emptyset.$$

Also gilt $\varphi \notin \mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cap\}\Big)$. □

Damit gilt folgender Satz:

**Satz 3.12**
$\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cap\}\Big)$ *ist* $\leq_{\mathrm{m}}^{\log}$*-vollständig für* P.

*Beweis.* Die Aussage folgt aus den Lemmata 3.10 und 3.11. □

### 3.3.2 $\mathbf{CSP}\Big([\mathbb{N}], \{=, \cap\}\Big)$

Wir zeigen für $\mathrm{CSP}\Big([\mathbb{N}], \{=, \cap\}\Big)$ die Zugehörigkeit zu P.

**Lemma 3.13**
*Es gilt* $\mathrm{CSP}\Big([\mathbb{N}], \{=, \cap\}\Big) \in \mathrm{P}$.

*Beweis.* Gemäß Lemma 2.5 ist es hinreichend, $\mathrm{CSP}'\big([\mathbb{N}], \{=, \cap\}\big) \in \mathrm{P}$ zu zeigen. Sei daher $\varphi$ eine $\{\cap\}$-Formel, deren Atome die Form $A \cap B = C$ haben.
Wir geben einen iterativen Algorithmus an, der $\mathrm{CSP}\Big([\mathbb{N}], \{=, \cap\}\Big)$ entscheidet. Betrachte die Abbildung $\alpha : \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi \to [\mathbb{N}]$, die vermöge $\alpha(X) = \emptyset$ für $X \in \mathrm{Var}_\varphi$ und $\alpha(C) = C$ für $C \in \mathrm{Const}_\varphi$ definiert sei. Der folgende Algorithmus verändert die Werte der Abbildung:

1. Wende für jedes Atom $X \cap Y = Z$ die folgenden Regeln an:
   (a) Setze zunächst $\alpha(Z) = \alpha(Z) \cup (\alpha(X) \cap \alpha(Y))$ und daraufhin dann $\alpha(Z) = [\min\big(\alpha(Z)\big), \max\big(\alpha(Z)\big)]$.
   (b) Setze zuerst $\alpha(X) = \alpha(X) \cup \alpha(Z)$ und dann $\alpha(X) = [\min\big(\alpha(X)\big), \max\big(\alpha(X)\big)]$ und analog für $Y$.
   (c) Falls $\alpha(C)$ für eine Konstante $C$ geändert wurde, gib 0 zurück.
2. Hat sich in Schritt 1 eine Änderung eines Wertes $\alpha(X)$ für ein $X \in \mathrm{Var}_\varphi$ ergeben, so führe Schritt 1 erneut aus.
3. Gib 1 zurück.

In jeder Ausführung von Schritt 1 mit Ausnahme der letzten wird der Wert $\alpha(X)$ einer Variablen $X \in \mathrm{Var}_\varphi$ geändert. Induktiv lässt sich zeigen, dass $\alpha(X)$ für jedes $X \in \mathrm{Var}_\varphi$ zu jedem Zeitpunkt der Ausführung des Algorithmus einen Wert $[a, b]$ besitzt, wobei $a, b$ Intervallgrenzen von Konstanten der Formel $\varphi$ sind oder $[a, b] = \emptyset$ gilt:
Der Induktionsanfang ist klar. Es werde nun der Wert $\alpha(X)$ für ein $X \in \mathrm{Var}_\varphi$ verändert. Wir unterscheiden zwei Fälle:

1. Durch Anwendung der Regel 1a auf $X = Y \cap Z$ für $Y, Z \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$ wird $\alpha(X)$ geändert. Nach Induktionsannahme haben $\alpha(X), \alpha(Y), \alpha(Z)$ vor Anwendung der Regeln Intervallgrenzen, die auch in Konstanten als Intervallgrenzen auftreten. Dann gilt dies auch für $\alpha(Y) \cap \alpha(Z)$ und damit auch für $\alpha(X)$ nach Anwendung der Regeln.

2. Der Fall $Z = X \cap Y$ für $Y, Z \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$ kann analog behandelt werden.

Da es nur linear viele Intervallgrenzen von Konstanten gibt und die Werte $\alpha(X)$ für $X \in \mathrm{Var}_\varphi$ monoton wachsen, führt der Algorithmus nur polynomiell oft den Schritt 1 aus. Der Algorithmus arbeitet somit in Polynomialzeit.

Wir zeigen nun, dass der Algorithmus genau dann 1 zurückgibt, wenn es eine erfüllende Belegung der Variablen von $\varphi$ gibt.

„⇒“: Es gebe der Algorithmus 1 zurück. Wir zeigen, dass dann $\alpha_{|\mathrm{Var}_\varphi}$ mit den Werten bei der Terminierung des Algorithmus eine erfüllende Belegung darstellt. Angenommen, dies ist nicht so. Dann gibt es ein Atom $Z = X \cap Y$ mit

$$\alpha(X) \cap \alpha(Y) \neq \alpha(Z).$$

Wir unterscheiden zwei Fälle.

1. $\exists x \in \big(\alpha(X) \cap \alpha(Y)\big) - \alpha(Z)$. In dieser Situation hätte der Algorithmus jedoch nicht mit Rückgabe 1 terminiert, sondern die Regel 1a auf $\alpha(Z)$ angewendet.
2. $\exists x \in \alpha(Z) - \big(\alpha(X) \cap \alpha(Y)\big)$. Hier hätte der Algorithmus statt zu terminieren die Regel 1b angewendet.

Folglich ist $\alpha$ eine erfüllende Belegung der Variablen.

„⇐“: Sei $\beta : \mathrm{Var}_\varphi \to [\mathbb{N}]$ eine erfüllende Belegung der Variablen von $\varphi$. Wir schreiben weiterhin $\beta(C)$ für den Wert $C$ einer Konstanten $C$. Wir zeigen, dass zu jedem Zeitpunkt der Ausführung des Algorithmus $\alpha(X) \subseteq \beta(X)$ für alle $X \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$ gilt. Zu Beginn der Ausführung gilt dies offenbar. Für ein beliebiges Atom $X \cap Y = Z$ schreiben wir $U =_{\mathrm{def}} \alpha(X)$, $V =_{\mathrm{def}} \alpha(Y)$ und $W =_{\mathrm{def}} \alpha(Z)$ für die aktuellen Werte und betrachten, was passiert wenn eine der Regeln angewendet wird. Wir schreiben dabei $U', V', W'$ für die Werte von $\alpha(X), \alpha(Y), \alpha(Z)$ nach der Ausführung der jeweiligen Regel.

1. Es werde die Regel 1a angewendet. Es gilt

   $$U \subseteq \alpha(X) \qquad V \subseteq \alpha(Y) \qquad W \subseteq \alpha(Z).$$

   Weiter gilt dann

   $$W' = W \cup (U \cap V) \subseteq \beta(Z) \cup \big(\beta(X) \cap \beta(Y)\big) \subseteq \beta(Z).$$

2. Es werde die Regel 1b angewendet. Wir betrachten lediglich die Regel für $X$. Es gilt

   $$U \subseteq \alpha(X) \qquad W \subseteq \alpha(Z).$$

   Weiter gilt dann wegen $\beta(Z) \subseteq \beta(X)$

   $$U' = U \cup W \subseteq \beta(X) \cup \beta(Z) \subseteq \beta(X).$$

Insbesondere gilt also für jede Konstante $C$

$$\alpha(C) \subseteq \beta(C)$$

zu jedem Zeitpunkt der Ausführung des Algorithmus, weswegen nicht 0 zurückgegeben wird. Da der Algorithmus wie oben begründet nach vielen polynomiell vielen Rechenschritten terminiert, gibt er den Wert 1 zurück.

□

**Satz 3.14**
$\mathrm{CSP}\Big([\mathbb{N}], \{=, \cap\}\Big)$ *ist* $\leq_{\mathrm{m}}^{\log}$*-vollständig für* P.

*Beweis.* Die Aussage folgt aus Lemma 3.13 und Lemma 3.11. □

## 3.4 Schnitt und Vereinigung

In diesem Abschnitt wird die $\leq_{\mathrm{m}}^{\log}$-Vollständigkeit der Probleme $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup, \cap\}\Big)$ und $\mathrm{CSP}\Big([\mathbb{N}], \{=, \cup, \cap\}\Big)$ für NP gezeigt.

### 3.4.1 $\mathbf{CSP}\Big(\mathcal{P}_{\mathbf{fin}}(\mathbb{N}), \{=, \cup, \cap\}\Big)$

Wir zeigen die $\leq_{\mathrm{m}}^{\log}$-Härte für NP durch eine Reduktion von 3SAT.

**Lemma 3.15**
$\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup, \cap\}\Big)$ *ist* $\leq_{\mathrm{m}}^{\log}$*-hart für* NP.

*Beweis.* Wir zeigen $\mathrm{3SAT} \leq_{\mathrm{m}}^{\log} \mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup, \cap\}\Big)$. Weil 3SAT $\leq_{\mathrm{m}}^{\log}$-vollständig für NP ist, folgt dann die Behauptung. Es sei $\psi$ eine beliebige aussagenlogische Formel in konjunktiver Normalform mit genau 3 Literalen pro Klausel und Variablen $x_1, \ldots, x_n$, also

$$\psi = \bigwedge_{i=1}^{r} (l_{i,1} \vee l_{i,2} \vee l_{i,3})$$

für Literale $l_{i,j} \in \{x_{i,j}, \neg x_{i,j}\}$ und $x_{i,j} \in \{x_1, \ldots, x_n\}$.

Wir konstruieren eine Formel $\varphi$, sodass

$$\varphi \in \mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup, \cap\}\Big) \Leftrightarrow \psi \in \mathrm{3SAT}$$

gilt. Dabei beschreibt die Menge $\{0\}$ den Wahrheitswert 0 und die Menge $\{0, 1\}$ den Wahrheitswert 1.

$$\begin{aligned}
\varphi = &\exists X_1 \exists Y_1 \exists Z_1 \ldots \exists X_n \exists Y_n \exists Z_n \; \exists L_{1,1} \exists L_{1,2} \exists L_{1,3} \ldots \exists L_{n,1} \exists L_{n,2} \exists L_{n,3} \\
&\bigwedge_{i=1}^{n} \Big( X_i \cup Y_i = \{0,1\} \wedge Z_i \cup \{0\} = X_i \Big) \wedge \\
&\bigwedge_{i \in \{1,\ldots,r\}, j \in \{1,2,3\}, s \in \{1,\ldots,n\}, l_{i,j} = \neg x_s} \Big( L_{i,j} \cup X_s = \{0,1\} \wedge L_{i,j} \cap X_s = \{0\} \Big) \wedge \\
&\bigwedge_{i \in \{1,\ldots,r\}, j \in \{1,2,3\}, s \in \{1,\ldots,n\}, l_{i,j} = x_s} \Big( L_{i,j} = X_s \Big) \wedge \bigwedge_{i \in \{1,\ldots,r\}} \bigcup_{j=1}^{3} \Big( L_{i,j} = \{0,1\} \Big)
\end{aligned}$$

Die zweite Zeile der Formel bringt zum Ausdruck, dass für $i = 1, \ldots, n$ unter einer erfüllenden Variablenbelegung $\alpha$

$$\alpha(X_i) \in \{\{0\}, \{0,1\}\}$$

gelten muss.
Die dritte Zeile fordert, dass die für Literale $l_{i,j} = \neg x_s$ stehenden Variablen $L_{i,j}$ einen Wert aus $\{\{0,1\}, \{0\}\}$ annehmen und genau dann den Wert $\{0,1\}$ unter einer erfüllenden Variablenbelegung erhalten, wenn $X_s$ diesen Wert nicht hat.
In der vierten Zeile wird ausgedrückt, dass jede für ein Literal $l_{i,j} = x_s$ stehende Variable $L_{i,j}$ den Wert von $X_s$ erhält und dass für jede Klausel mindestens eine Variable $L_{i,j}$ den Wert $\{0,1\}$ hat.

Es kann $f$ in logarithmischem Raum berechnet werden, da zu jedem Zeitpunkt der Berechnung lediglich konstant viele Variablen sowie die Anzahl der Variablen und Klauseln in $\psi$ zu speichern sind.

Für eine Belegung der Variablen $x_i$ mit Werten 0 oder 1 definiere eine Belegung der Variablen $X_i$ wie folgt:
Falls $x_i = 1$, setze $X_i = \{0, 1\}$, andernfalls setze $X_i = \{0\}$. Dann ist die Belegung der Variablen $x_i$ genau dann erfüllend, wenn die von der beschriebenen Belegung der Variablen $X_i$ induzierte Termbelegung erfüllend ist. □

Damit lässt sich folgender Satz beweisen:

**Satz 3.16**
$\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \cup, \cap\}\Big)$ *ist* $\leq_{\mathrm{m}}^{\log}$*-vollständig für* NP.

*Beweis.* Die Aussage folgt aus Lemma 3.4 und Bemerkung 3.2. □

### 3.4.2 $\mathbf{CSP}\Big([\mathbb{N}], \{=, \cup, \cap\}\Big)$

Mit Hilfe von Lemma 3.8 lässt sich der folgende Satz zeigen:

**Satz 3.17**
$\mathrm{CSP}\Big([\mathbb{N}], \{=, \cup, \cap\}\Big)$ *ist* $\leq_{\mathrm{m}}^{\log}$*-vollständig für* NP.

*Beweis.* Die Härte folgt aus Lemma 3.8 sowie Lemma 2.3. Aus Bemerkung 3.2 folgt $\mathrm{CSP}\Big([\mathbb{N}], \{=, \cup, \cap\}\Big) \in$ NP. □

## 3.5 Mengen-Differenz

Dieser Abschnitt dient der Betrachtung von CSPs, in denen ausschließlich Mengenoperationen, in jedem Fall aber die Differenz zwischen zwei Mengen, zugelassen sind.

Wenn wir CSPs über $\mathcal{P}_{\text{fin}}(\mathbb{N})$ betrachten, gestaltet sich die Situation sehr einfach, wie es die folgende Bemerkung darlegt.

**Bemerkung 3.18**
*Betrachte* $\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \text{O}\big)$ *für* $O \subseteq \{-, \cap, \cup\}$ *mit* $- \in O$. *Laut Bemerkung 3.2 ist dieses Problem in* NP. *Nach Lemma 3.15 ist* $\text{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, \cup, \cap\}\Big)$ $\leq_{\text{m}}^{\log}$*-hart für* NP. *Lemma 2.3 liefert die* $\leq_{\text{m}}^{\log}$*-Härte von* $\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, \cup, \cap, -\}\big)$. *Folglich erhalten wir mit Lemma 2.6 die* $\leq_{\text{m}}^{\log}$*-Vollständigkeit von* $\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \text{O}\big)$ *für beliebiges* $O \subseteq \{-, \cap, \cup\}$ *mit* $- \in O$ *für* NP.

Etwas schwieriger gestaltet sich die Situation für CSPs über $[\mathbb{N}]$. Aus Lemma 3.2 erhalten wir wiederum die Zugehörigkeit zu NP. Aber auch die $\leq_{\text{m}}^{\log}$-Härte für NP von

$$\text{CSP}\big([\mathbb{N}], \{=\} \cup \text{O}\big), \quad \text{O} \subseteq \{-, \cup, \cap\} \text{ mit } - \in \text{O},$$

lässt sich zeigen. Wir gehen dazu ähnlich wie in Lemma 3.8 vor.

**Satz 3.19**
$\text{CSP}\big([\mathbb{N}], \{=\} \cup \text{O}\big)$ *für* $O \subseteq \{-, \cap, \cup\}$ *mit* $- \in O$ *ist* $\leq_{\text{m}}^{\log}$*-vollständig für* NP.

*Beweis.* Wie schon bemerkt genügt es, die $\leq_{\text{m}}^{\log}$-Härte für NP zu zeigen. Wegen Lemma 2.3 ist es hinreichend, $3\text{SAT} \leq_{\text{m}}^{\log} \text{CSP}\big([\mathbb{N}], \{=, -\}\big)$ zu beweisen.

Sei $H$ eine aussagenlogische Formel in konjunktiver Normalform mit genau 3 Variablen pro Klausel und Variablen $x_1, \ldots, x_n$, also

$$H = \bigwedge_{i=1}^{m} \big( \bigvee_{j=1}^{3} l_{i,j} \big),$$

wobei die $l_{i,j}$ Literale sind, d.h. es existiert $s : \{(i,j) \mid i \in \{1, \ldots, m\}, j \in \{1,2,3\}\}$ mit $l_{i,j} \in \{x_{s(i,j)}, \neg x_{s(i,j)}\}$.

Wir beschreiben eine Formel $\varphi$, die genau dann wahr ist, wenn $H$ erfüllbar ist. Dabei steht die Menge [0] für den Wahrheitswert 0, die Menge [2] für den Wahrheitswert 1. Wir fordern daher in unserer Formel, dass die CSP-Variablen $X_i$, die für die Variablen aus $H$ stehen, Werte aus $\{[0], [2]\}$ annehmen, wobei das entsprechende $X_i'$ den jeweils anderen Wert annimmt (Zeile 2). Ferner verwenden wir in $\varphi$ Variablen $L_{i,j}$, welche für die einzelnen Literale stehen.
Zuletzt (Zeile 4) bleibt noch zu fordern, dass für alle $i = 1, \ldots, m$ in der $i$-ten Klausel ein Literal $L_{i,j}$ den Wert [2] hat.

$$\exists X_1 \exists X_1' \ldots \exists X_n \exists X_n' \exists L_{1,1} \exists L_{1,2} \exists L_{1,3} \ldots \exists L_{m,1} \exists L_{m,2} \exists L_{m,3}$$

$$\bigwedge_{i=1,\ldots,n} \Big( \big( [0,2] - X \big) - X' = \emptyset \ \wedge \ X - ([0,2] - [1]) = \emptyset \wedge X' - ([0,2] - [1]) = \emptyset \Big)$$

$$\bigwedge_{j=1,\ldots,m} \bigwedge_{k=1}^{3} L_{j,k} = \begin{cases} X_{s(j,k)} & \text{falls } l_{j,k} = x_{s(j,k)} \\ X'_{s(j,k)} & \text{falls } l_{j,k} = \neg x_{s(j,k)} \end{cases}$$

$$\bigwedge_{j=1,\ldots,m} \Big( \big( [2] - L_{i,1} \big) - L_{i,2} \Big) - L_{i,3} = \emptyset.$$

□

## 3.6 Übersicht und Fazit

Folgende Tabellen liefert eine Übersicht über die in diesem Kapitel gewonnenen Resultate. Die erste Tabelle behandelt die CSPs über $\mathcal{P}_{\text{fin}}(\mathbb{N})$.

| $\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \text{O}\big)$ mit $O =$ | $\leq_{\text{m}}^{\log}$-hart für | enthalten in |
|---|---|---|
| $\emptyset$ | L | L, 3.4 |
| $\{\cup\}$ | P, 3.6 | P, 3.5 |
| $\{\cap\}$ | P, 3.11 | P, 3.10 |
| $\{\cup, \cap\}$ | NP, 3.15 | NP, 3.2 |
| $\{-\}$ | NP, 3.18 | NP, 3.2 |
| $\{-, \cup\}$ | NP, 3.18 | NP, 3.2 |
| $\{-, \cap\}$ | NP, 3.18 | NP, 3.2 |
| $\{-, \cup, \cap\}$ | NP, 3.18 | NP, 3.2 |

Die nachfolgende Tabelle gibt Aufschluss über die CSPs über $[\mathbb{N}]$.

| $\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \text{O}\big)$ mit $O =$ | $\leq_{\text{m}}^{\log}$-hart für | enthalten in |
|---|---|---|
| $\emptyset$ | L | L, 3.4 |
| $\{\cup\}$ | NP, 3.8 | NP, 3.2 |
| $\{\cap\}$ | P, 3.11 | P, 3.13 |
| $\{\cup, \cap\}$ | NP, 3.17 | NP, 3.17 |
| $\{-\}$ | NP, 3.19 | NP, 3.2 |
| $\{-, \cup\}$ | NP, 3.19 | NP, 3.2 |
| $\{-, \cap\}$ | NP, 3.19 | NP, 3.2 |
| $\{-, \cup, \cap\}$ | NP, 3.19 | NP, 3.2 |

Für jedes der Probleme ist somit die $\leq_{\text{m}}^{\log}$-Vollständigkeit für eine der Komplexitätsklassen $\text{L}, \text{P}, \text{NP}$ gezeigt.

Ferner ist ersichtlich, dass wir in beiden Situationen mit Ausname von den CSPs über $\{\cup\}$ stets die gleichen Resultate erhalten. Die genannte Ausnahme bildet in dieser Arbeit den einzigen Fall, für den ein CSP über $[\mathbb{N}]$ schwerer zu entscheiden ist als das korrespondierende CSP über $\mathcal{P}_{\text{fin}}(\mathbb{N})$[9]. In Abschnitt 4 werden wir auf einige Beispiele stoßen, in denen das Gegenteil der Fall ist.

[9] Diese Aussage gilt freilich nur, sofern $\text{P} \neq \text{NP}$ gilt.

# 4 CSPs mit Arithmetischen Operationen

Bevor wir uns einzelnen konkreten Problemen zuwenden, zeigen wir eine obere Schranke für alle in dieser Arbeit definierten CSPs:

**Satz 4.1**
*Sei $M \in \{\mathcal{P}_{\text{fin}}(\mathbb{N}), [\mathbb{N}]\}$ und $O \subseteq \{+, \times, \cup, \cap, -\}$. Dann gilt*

$$\text{CSP}\Big(\text{M}, \{=\} \cup \text{O}\Big) \in \Sigma_1.$$

*Beweis.* Die Menge

$$\{(\varphi, \alpha) \mid \varphi \in \text{CSP}\Big(\text{M}, \{=\} \cup \text{O}\Big),\ \alpha \text{ ist eine erfüllende Variablenbelegung für } \varphi\}$$

ist entscheidbar, da jede der endlich vielen Variablen mit einer endlichen Menge belegt wird und für jedes Atom überprüft werden kann, ob es von $\alpha$ erfüllt wird. Also ist das Problem $\text{CSP}\Big(\text{M}, \{=\} \cup \text{O}\Big)$ als Projektion einer entscheidbaren Menge in $\Sigma_1$. □

## 4.1 Addition

Dieser Abschnitt dient der Untersuchung zweier Probleme: $\text{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +\}\Big)$ und $\text{CSP}\Big([\mathbb{N}], \{=, +\}\Big)$. Dabei werden wir für $\text{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +\}\Big)$ die NP-Härte und die Zugehörigkeit zu NEXP sowie für $\text{CSP}\Big([\mathbb{N}], \{=, +\}\Big)$ die NP-Vollständigkeit zeigen.

Wir zeigen zunächst, dass im Falle der Existenz einer erfüllenden Variablenbelegung für eine konkrete Formel $\varphi$ auch eine erfüllende Belegung existiert, deren Werte einer oberen Schranke genügen:

**Lemma 4.2**
*Es sei $\varphi \in \text{CSP}'\Big(\text{M}, \{=, +\}\Big)$ für $M \in \{\mathcal{P}_{\text{fin}}(\mathbb{N}), [\mathbb{N}]\}$. Weiterhin sei*

$$x = \max\{\bigcup_{C \in \text{Const}_\varphi} C\}^{10}$$

*und $n = |\varphi|$. Dann gibt es eine erfüllende Variablenbelegung $\alpha$ für $\varphi$ mit*

$$\forall X \in \text{Var}_\varphi : \quad \max(\alpha(\text{X})) \leq \text{x}2^{\text{n}}.$$

*Beweis.* Der folgende nichtdeterministische Algorithmus konstruiert eine totale Funktion $\alpha : \text{Var}_\varphi \cup \text{Const}_\varphi \to \text{M}$, wobei $\alpha(C) =_{\text{def}} C$ für $C \in \text{Const}_\varphi$ sei. Es sei $\alpha(X)$ für $X \in \text{Var}_\varphi$ zunächst nicht definiert.

1. flag = true
2. while flag
   (a) flag = false
   (b) Für jede Variable $X$, für die eine Gleichung $X = Y + Z$ existiert, sodass $\alpha(Y)$ und $\alpha(Z)$ bereits definiert sind, $\alpha(X)$ hingegen noch nicht definiert ist, definiere

$$\alpha(X) =_{\text{def}} \alpha(Y) + \alpha(Z)$$

   und setze flag = true.

[10] Es sei dabei $\max(\emptyset) =_{\text{def}} -\infty$ und $-\infty \leq \zeta$ für $\zeta \in \mathbb{N} \cup \{-\infty\}$, sowie $\zeta \cdot (-\infty) = -\infty$.

(c) Für jede Variable $X$ mit nicht definiertem $\alpha(X)$, für die Variablen $Z_1, Z_2$ mit $X + Z_1 = Z_2$ existieren, wobei $\alpha(Z_2)$ definiert und $\alpha(Z_2) \neq \emptyset$ ist, rate nichtdeterministisch eine Menge $S \in M$, sodass $\max\{S\} \leq \max(\alpha(Z_2)$ gilt, definiere $\alpha(X) =_{\text{def}} S$ und setze flag = true.

3. Für alle Variablen $X$ mit nicht definiertem $\alpha(X)$ definiere $\alpha(X) =_{\text{def}} \emptyset$.

4. Falls $\alpha$ eine erfüllende Belegung ist, gib $\alpha$ zurück.

Gibt der Algorithmus auf einem Rechenweg eine Belegung zurück, so ist dies offenbar eine erfüllende Belegung.

Wir zeigen nun, dass der Algorithmus auf mindestens einem Rechenweg eine Belegung zurückgibt. Es terminiert der Algorithmus auf jedem Rechenweg, da nur dann ein weiterer Schleifendurchlauf durchgeführt wird, wenn im letzten Durchlauf der Wert einer Variable definiert wurde. Es gibt also höchstens $|\text{Var}_\varphi|$ Schleifendurchläufe.

Sei $\beta$ eine erfüllende Termbelegung für $\varphi$. Wir zeigen induktiv, dass vor jedem Schleifendurchlauf ein Rechenweg existiert, sodass für die auf diesem Rechenweg konstruierte Belegung $\alpha$ das Folgende gilt:

$$\forall X \in \text{Var}_\varphi : \alpha(\text{X}) \text{ nicht definiert } \vee \alpha(\text{X}) = \beta(\text{X}).$$

Vor dem ersten Schleifendurchlauf gilt die Bedingung offenbar. Angenommen, die Bedingung ist vor einem beliebigen Schleifendurchlauf erfüllt.
Sei $X$ eine Variable, deren Wert $\alpha(X)$ in Schritt 2b definiert wird. Dann gilt nach Induktionsvoraussetzung $\alpha(Y) = \beta(Y)$ und $\alpha(Z) = \beta(Z)$. Da $\beta$ erfüllend ist, gilt $\beta(X) = \beta(Y) + \beta(Z)$ und somit gilt nach Ausführung von Schritt 2b

$$\alpha(X) = \beta(X).$$

Für eine Variable $X$, deren Wert $\alpha(X)$ in Schritt 2c definiert wird, gilt $\alpha(Z_2) \neq \emptyset$ und nach Induktionsvoraussetzung $\alpha(Z_2) = \beta(Z_2)$. Da $\beta$ erfüllend ist und für die Addition $A + B = C$ mit $A, B, C \in M$ und $C \neq \emptyset$

$$\max(C) \geq \max(A \cup B)$$

gilt, folgt $\max(\beta(X)) \leq \max(\beta(Z_2))$. Also gibt es einen Rechenweg mit $\alpha(X) = \beta(X)$ nach der Ausführung von Schritt 2c.

Beim Erreichen von Schritt 3 werden alle Atome $X + Y = Z$, für die $\alpha(X)$, $\alpha(Y)$ und $\alpha(Z)$ definiert sind, von $\alpha$ erfüllt.
Sei daher $X + Y = Z$ ein Atom, sodass einer der Werte $\alpha(X)$, $\alpha(Y)$ und $\alpha(Z)$ noch nicht definiert ist.
Wäre $\alpha(Z)$ definiert und $\alpha(Z) \neq \emptyset$, so wären wegen Schritt 2c auch $\alpha(X)$ und $\alpha(Y)$ definiert. Falls $\alpha(Z) = \emptyset$, so ist durch das Setzen von $\alpha(X) = \emptyset$ bzw. $\alpha(Y) = \emptyset$ das Atom $X + Y = Z$ erfüllt. Sei nun $\alpha(Z)$ noch nicht definiert, dann ist auch o.B.d.A. $\alpha(X)$ noch nicht definiert (wegen Schritt 2b). Durch das Setzen von $\alpha(X) = \alpha(Z) = \emptyset$, gilt $\alpha(X) + \alpha(Y) = \alpha(Z)$. Daher ist nach Schritt 3 jede Gleichung aus $\varphi$ erfüllt. $\alpha$ ist folglich eine erfüllende Belegung.

Für jede Variable $X$ wird genau einmal der Wert $\alpha(X)$ definiert. Es seien die Variablen mit $X_1, \ldots, X_{|\text{Var}_\varphi|}$ so bezeichnet, dass für $i < j$ der Wert $\alpha(X_i)$ vor $\alpha(X_j)$ definiert wird. Direkt aus dem Algorithmus folgt

$$\max(\alpha(X_{i+1})) \leq 2 \cdot \max\Big(\bigcup_{j \leq i} \alpha(X_j)\Big).$$

Daher lässt sich durch eine einfache Induktion

$$\forall i: \quad \max(\alpha(X_i)) \leq x2^i$$

zeigen. Aus $|\mathrm{Var}_\varphi| \leq$ n folgt die Behauptung. □

Mit diesem Resultat ist es hinreichend, alle Variablenbelegungen mit Werten $\leq x \cdot 2^n$ zu raten, zu testen, ob die jeweilige Belegung erfüllend ist und dann einen entsprechenden Wert zurückzugeben. Dies zeigt die Entscheidbarkeit der genannten Probleme. Eine Aussage hinsichtlich der Komplexität liefert der folgende Satz.

**Satz 4.3**
*Es gilt*

1. $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +\}\Big) \in \mathrm{NEXP}$
2. $\mathrm{CSP}\Big([\mathbb{N}], \{=, +\}\Big) \in \mathrm{NP}$.

*Beweis.* Gemäß Lemma 2.5 genügt es, die beiden Aussagen für $\mathrm{CSP}'\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +\}\Big)$ bzw. $\mathrm{CSP}'\Big([\mathbb{N}], \{=, +\}\Big)$ zu zeigen.
1.: Gemäß Lemma 4.2 entscheidet der folgende Algorithmus $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +\}\Big)$.

- Eingabe: $\varphi$
- Bestimme $n = |\varphi|$ und die größte Zahl $x$, die in einer von einem Konstantensymbol codierten Menge auftritt.
- Rate $\alpha(X) \subseteq \{k \in \mathbb{N} \mid k \leq x \cdot 2^n\}$ für alle $X \in \mathrm{Var}_\varphi$.
- Prüfe, ob $\alpha$ erfüllend ist und gib entsprechend 0 oder 1 zurück.

Die Mengen $\alpha(X)$ enthalten höchstens $x \cdot 2^n$ Zahlen linearer Länge. Das Raten solcher Mengen, sowie das Überprüfen, ob eine gegebene Variablenbelegung erfüllend ist, ist damit offenbar in einer Laufzeit von $2^{p(n)}$ für ein geeignetes Polynom $p$ möglich.

2. Es kann nahezu der gleiche Algorithmus verwendet werden:

- Eingabe: $\varphi$
- Bestimme $n = |\varphi|$ und die größte Zahl $x$, die in einer von einem Konstantensymbol codierten Menge auftritt.
- Rate $\alpha(X) = [k_1, k_2]$ mit $k_1, k_2 \leq x \cdot 2^n$ für alle $X \in \mathrm{Var}_\varphi$.
- Prüfe, ob $\alpha$ erfüllend ist und gib entsprechend 0 oder 1 zurück.

Hier sind lediglich $2|\mathrm{Var}_\varphi|$ Zahlen $\leq x \cdot 2^n$, d.h. mit linearer Länge, zu raten, was in nichtdeterministischer Polynomialzeit möglich ist. Für ein gegebenes Atom kann wegen der Längenbeschränkung der Intervallgrenzen leicht in linearer Laufzeit entschieden werden, ob es erfüllt ist. Also arbeitet der Algorithmus in nichtdeterministischer Polynomialzeit. □

**Bemerkung 4.4**
*Der zweite Teil von Satz 4.3 lässt sich auch einfacher zeigen. Statt zu zeigen, dass es eine obere Schranke $S$ gibt, für welche es hinreichend ist, die Intervallgrenzen aus der Menge $[0, S]$ auszuwählen, könnte man auch eine Reduktion von* $\mathrm{CSP}\Big([\mathbb{N}], \{=, +\}\Big)$ *auf das* NP*-Problem* ILP *durchführen.*
*Ein derartiger Beweis für eine noch stärkere Aussage, nämlich* $\mathrm{CSP}\Big([\mathbb{N}], \{=, +, \cap\}\Big) \in \mathrm{NP}$*, findet sich im Beweis des Satzes 4.22.*
*Das Resultat* $\mathrm{CSP}\Big([\mathbb{N}], \{=, +\}\Big) \in \mathrm{NP}$ *lässt sich jedoch sehr leicht aus Lemma 4.2 , welches für den Beweis des ersten Teils von Satz 4.3 ohnehin benötigt wird, ableiten. Aus diesem Grunde lag es nahe,* $\mathrm{CSP}\Big([\mathbb{N}], \{=, +, \cap\}\Big) \in \mathrm{NP}$ *auch auf die geschilderte Weise zu zeigen.*

Nun sind für beide Probleme untere Schranken zu finden. Es lässt sich jeweils die $\leq_{\mathrm{m}}^{\log}$-Härte für NP mittels einer Reduktion von einer Variante von SOS zeigen.
Die von dieser Reduktion konstruierten Formeln sind dabei so geartet, dass allen auftretenden Variablen unter einer erfüllenden Belegung ausschließlich einelementige Mengen zugewiesen werden. Es ist also im Fall von $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +\}\Big)$ nicht nötig, die gegebenen Möglichkeiten auszuschöpfen. Es liegt daher die Vermutung nahe, dass die untere Schranke nicht scharf ist.
Für $\mathrm{CSP}\Big([\mathbb{N}], \{=, +\}\Big)$ liefert das genannte Resultat hingegen die NP-Vollständigkeit.

Dies stellt ein Indiz dafür dar, dass an dieser Stelle die CSPs über endlichen Teilmengen der natürlichen Zahlen schwerer als die entsprechenden CSPs über endlichen Intervallen sind. In Abschnitt 3.2 hatten wir die umgekehrte Situation beobachtet.

Wir definieren nun die folgende Variante von SOS.

**Definition 4.5**
Es sei

$$\mathrm{MSOS} = \{\langle x_1, \ldots, x_n, b\rangle\} \mid \exists a_1, \ldots, a_n \in \mathbb{N} : \sum_{i \in I} a_i x_i = b\}.$$

**Lemma 4.6 ([BGSW05])**
MSOS *ist* $\leq_{\mathrm{m}}^{\log}$*-vollständig für* NP*.*

Dies ermöglicht uns, den folgenden Satz zu beweisen.

**Satz 4.7**
$\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +\}\Big)$ *und* $\mathrm{CSP}\Big([\mathbb{N}], \{=, +\}\Big)$ *sind* $\leq_{\mathrm{m}}^{\log}$*-hart für* NP*.*

*Beweis.* Sei $M \in \{\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), [\mathbb{N}]\}$. Wir geben eine $\leq_{\mathrm{m}}^{\log}$-Reduktion von MSOS auf das Problem $\mathrm{CSP}\Big(\mathrm{M}, \{=, +\}\Big)$ an.

Es sei eine Menge $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{N}$ und ein $b \in \mathbb{N}$ gegeben.
Wir konstruieren einen Satz $\varphi$, der genau dann wahr ist, wenn $\langle x_1, \ldots, x_n, b\rangle \in \mathrm{MSOS}$ gilt.

Dazu nehmen wir an, dass die $x_i$ und $b$ in der MSOS-Instanz in Binärdarstellung codiert sind[11]. Bei der Konstruktion des Satzes $\varphi$ werden wir ebenfalls alle auftretenden Zahlen binär darstellen.

[11] Andere typische Codierungen wie etwa die dyadische Codierung lassen sich in logarithmischem Raum in die Binärdarstellung umrechnen.

Das Prinzip unserer Konstruktion sieht so aus, dass die Werte $a_i$ mit einem Existenzquantor in dem Satz $\varphi$ „geraten“ werden und mittels der „Shift-And-Add“-Technik daraufhin die Werte $a_i x_i$ berechnet werden, wobei wir direkt auf die Bits der Zahl $x_i$ zugreifen können.
Zunächst beschreiben wir einen Satz, der bewirkt, dass eine Variable unter einer gültigen Termbelegung den Wert $\{a_i x_i\}$ annimmt, wobei $a_i$ geraten wird. Sei dazu $b_{i,m_i} b_{i,m_i-1} \dots b_{i,0}$ die Binärdarstellung von $x_i$.
Dann gilt

$$a_i x_i = \sum_{j=0}^{m_i} a_i b_{i,j} 2^j = 2 \cdot \Big( \dots \Big( 2\big(2 a_i b_{i,m_i} + a_i b_{i,m_i-1}\big) + a_i b_{i,m_i-2} \Big) \dots \Big) + a_i b_{i,0}. \qquad (*)$$

Betrachte folgenden Satz:

$$\begin{aligned} \psi_i = & \exists A_i \exists R_{i,0} \exists R_{i,1} \dots \exists R_{i,m_i} \left( R_{i,m_i} = \begin{cases} \{0\} & b_{i,m_i} = 0 \\ A_i & \text{sonst} \end{cases} \right) \wedge \\ & \bigwedge_{m_i - 1 \geq j \geq 0} \left( R_{i,j} = (R_{i,j+1} + R_{i,j+1}) + \begin{cases} \{0\} & b_{i,j} = 0 \\ A_i & \text{sonst} \end{cases} \right). \end{aligned}$$

Für jede erfüllende Termbelegung $\alpha$, die jeder Variablen eine einelementige Menge zuweist, gilt gemäß $(*)$

$$\alpha(R_{i,0}) = \{a_i x_i\},$$

sofern $a_i$ das eindeutige Element in $\alpha(A_i)$ ist.

Es sei $\varphi$ die Formel, die aus

$$\bigwedge_{i=1}^{n} \psi_i \wedge \big( \sum_{i=1}^{n} R_{i,0} = \{b\} \big)$$

entsteht, wenn alle Existenzquantoren an den Anfang der Formel geschrieben werden.

Jede den Satz $\varphi$ erfüllende Termbelegung $\alpha$ belegt jede Variable mit einer einelementigen Menge: Andernfalls würde $\alpha$ auch mindestens ein $R_{i,0}$ mit einer nicht einelementigen Menge belegen, wodurch das Atom $\sum_{i=1}^{n} R_{i,0} = \{b\}$ nicht erfüllt sein könnte. Damit gilt dann für $M \in \{[\mathbb{N}], \mathcal{P}_{\text{fin}}(\mathbb{N})\}$

$$\varphi \in \mathrm{CSP}\big(\mathrm{M}, \{=, +\}\big) \Leftrightarrow \langle \mathrm{x}_1, \dots, \mathrm{x}_\mathrm{n}, \mathrm{b} \rangle \in \mathrm{MSOS}.$$

Ferner kann die skizzierte Funktion in logarithmischem Raum berechnet werden, weil zu jedem Zeitpunkt der Berechnung lediglich ein Wert aus $\{x_1, \dots, x_n, b\}$ sowie eine konstante Anzahl der Variablen von $\varphi$ gespeichert werden müssen. □

## 4.2 Multiplikation

Für die Multiplikation erhalten wir für die CSPs über endlichen Teilmengen der natürlichen Zahlen analoge Resultate wie für die Addition. Die Beweise gestalten sich jedoch etwas komplizierter. Das hat im Wesentlichen seine Ursache darin, dass die Multiplikation von nichtleeren Mengen wegen der Menge $\{0\}$ nicht monoton ist und es hierzu kein additives Pendant gibt.

Für $\mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$ ist die Situation eine gänzlich andere als für $\mathrm{CSP}\Big([\mathbb{N}], \{=, +\}\Big)$, da die Menge der endlichen Intervalle nicht abgeschlossen bezüglich $\times$ ist. In der Folge erhalten wir neben der NP-Härte von $\mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$ lediglich die Zugehörigkeit zu $\Sigma_3^{\mathrm{p}}$.

### 4.2.1 Obere Schranken

Wir beginnen mit dem Beweisen der oberen Schranken.

#### $\mathbf{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$

Zunächst betrachten wir die Multiplikation von Intervallen. Wir zeigen einige Eigenschaften, die das Entscheiden, ob eine $\mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$-Instanz wahr ist, wesentlich einfacher machen.

**Lemma 4.8**
*Es seien endliche Intervalle $A_1, \ldots, A_m, B_1, \ldots, B_n$ mit $\prod_{i=1}^{m} A_i = \prod_{i=1}^{n} B_i \neq \emptyset$ gegeben. Dann gilt*

$$\prod_{1 \leq i \leq m, |A_i|=1} A_i = \prod_{1 \leq i \leq n, |B_i|=1} B_i.$$

*Falls $\prod_{i=1}^{m} A_i \neq \{0\}$, so gilt auch*

$$\prod_{1 \leq i \leq m, |A_i| \neq 1} A_i = \prod_{1 \leq i \leq n, |B_i| \neq 1} B_i.$$

*Beweis.* Wir definieren $\{\alpha\} =_{\mathrm{def}} \prod_{1 \leq i \leq m, |A_i|=1} A_i$ und $\{\beta\} =_{\mathrm{def}} \prod_{1 \leq i \leq n, |B_i|=1} B_i$. Es ist keines der Intervalle $A_i, B_i$ leer. Daher gilt

$$\{\alpha\} \times \prod_{1 \leq i \leq m, |A_i| \geq 2} A_i = \{\beta\} \times \prod_{1 \leq i \leq n, |B_i| \geq 2} B_i.$$

Angenommen, es gilt $\alpha \neq \beta$. Wir nehmen o.B.d.A. $\alpha > \beta$ an. Dann ist für alle $x \in \prod_{i=1}^{m} A_i$ die Zahl $\alpha$ ein Teiler von $x$ und folglich ist $\alpha$ auch für jede Zahl $y \in \prod_{i=1}^{n} B_i$ ein Teiler von $y$. Wegen $\alpha > \beta$ gibt es eine Primzahlpotenz $p^e$, sodass $p^e \mid \alpha$ und $p^e \nmid \beta$ gilt.
In jedem Intervall $B_i$ mit $|B_i| \geq 2$ gibt es eine zu $p$ teilerfremde Zahl $b_i$: Falls der größte gemeinsame Teiler von $\min(B_i)$ und $p$ größer als 1 ist, so ist $\min(B_i) = rp$ für $r \in \mathbb{N}$. Es gilt dann $\gcd(r \cdot p + 1, p) = 1$.

Es gilt $b = \beta \cdot \prod_{1 \leq i \leq n, |B_i| \geq 2} b_i \in \prod_{i=1}^{n} B_i$ und $p^e \nmid b$. Folglich gilt auch $\alpha \nmid b$, ein Widerspruch.

Die zweite Behauptung folgt aus der ersten. □

**Lemma 4.9**
*Seien $A_1, \ldots, A_m, B_1, \ldots, B_n$ Intervalle mit jeweils mindestens zwei Elementen und*

$$\max(A_1) \leq \max(A_2) \leq \cdots \leq \max(A_m) \text{ sowie } \max(B_1) \leq \max(B_2) \leq \cdots \leq \max(B_n).$$

*Es gelte ferner $\prod_{i=1}^{m} A_i = \prod_{i=1}^{n} B_i$. Dann gilt $\max(A_m) = \max(B_n)$.*

*Beweis.* Es sei $L =_{\text{def}} \prod_{i=1}^{m} A_i$ und $R =_{\text{def}} \prod_{i=1}^{n} B_i$. Ferner seien die jeweils größten Elemente in $A_i$ bzw. $B_i$ mit $a_i$ bzw. $b_i$ bezeichnet.
Wegen $L = R$ gilt, dass das zweitgrößte Element in $L$ gleich dem zweitgrößten Element in $R$ ist, also

$$\max(L - \{\max(L)\}) = \max(R - \{\max(R)\}). \qquad (*)$$

Wir zeigen

$$\max(L - \{\max(L)\}) = \Big(\prod_{i=1}^{m-1} a_i\Big) \cdot (a_m - 1) = \max(L) - \prod_{i=1}^{m-1} a_i :$$

Sei $x \in L - \{\max(L)\}$ beliebig. Seien für $i = 1, \ldots, m$ Zahlen $x_i \in A_i$ gegeben, sodass $x = \prod_{i=1}^{m} x_i$ gilt. Wegen $x \neq \max(L)$ gibt es ein $j$, sodass $x_j < a_j$ ist. Dann gilt

$$x \leq \Big(\prod_{1 \leq i \leq m, i \neq j} a_i\Big) \cdot (a_j - 1) = \max(L) - \prod_{1 \leq i \leq m, i \neq j} a_i \leq \max(L) - \prod_{i=1}^{m-1} a_i.$$

Analog sieht man $\max(R - \{\max(R)\}) = \max(R) - \prod_{i=1}^{n-1} b_i$.
Aus $(*)$ und $\max(L) = \max(R)$ ergibt sich

$$\prod_{i=1}^{m-1} a_i = \prod_{i=1}^{n-1} b_i.$$

Daraus folgt

$$a_m = \frac{\max(L)}{\prod_{i=1}^{m-1} a_i} = \frac{\max(R)}{\prod_{i=1}^{n-1} b_i} = b_n.$$

□

Wenn also ein Produkt $M = C_1 \times \cdots \times C_k$ von Intervallen gegeben ist und Variablen $X_1, \ldots, X_n$ für $n \in \mathbb{N}$ so mit endlichen Intervallen belegt werden sollen, dass $\prod_{i=1}^{n} X_i = M$ gilt, dann muss für jedes $X_i$ und alle $x \in X_i$ die Bedingung $x \leq \max(\bigcup_{i=1}^{k} C_i)$ gelten.

Mit Hilfe dieser Eigenschaft können wir beweisen, dass es einen nichtdeterministischen Polynomialzeitalgorithmus gibt, der genau dann auf mindestens einem Rechenweg eine erfüllende Belegung für eine Formel $\varphi$ findet, wenn $\varphi \in \text{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$ ist. Der umseitig formulierte Satz präzisiert diese Behauptung.

**Satz 4.10**
*Es gibt einen nichtdeterministischen Polynomialzeitalgorithmus $\mathcal{A}$ mit folgenden Eigenschaften:*

- *$\mathcal{A}$ erhält als Eingabe eine $\{\times\}$-Formel $\varphi$, deren Konstanten endliche Intervalle sind.*
- *Auf jedem Rechenweg stoppt $\mathcal{A}$ entweder ohne Rückgabewert oder gibt ein Paar $(\psi, \alpha)$ zurück. Dabei ist $\psi$ eine $\{\times\}$-Formel, deren Konstanten endliche Intervalle sind, und es gilt*
$$\psi \in \mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big) \Rightarrow \varphi \in \mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big).$$
*$\alpha$ ist eine Variablenbelegung für $\psi$.*
- *Falls $\varphi \in \mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$, so gibt $\mathcal{A}$ auf mindestens einem Rechenweg eine erfüllende Variablenbelegung für $\psi$ zurück.*
- *Sei $x$ die größte in einer Konstanten von $\varphi$ auftretende Zahl. Für jedes zurückgegebene Paar $(\psi, \alpha)$ und für jede Variable $X \in \psi$ gilt* $\max(\alpha(X)) \leq x$.

*Beweis.* Wir beschreiben den Algorithmus $\mathcal{A}$. Dieser konstruiert eine Variablenbelegung $\alpha$, für die zu Beginn $\alpha(C) = C$ für $C \in \mathrm{Const}_\varphi$ gelte, wie folgt:

1. Rate nichtdeterministisch eine Menge $K' \subseteq \mathrm{Var}_\varphi$ und setze $\alpha(X) = \emptyset$ für alle Elemente $X$ von $K'$. Setze $K = K' \cup \{C \in \mathrm{Const}_\varphi \mid \mathrm{C} = \emptyset\}$[12].
2. Führe für jedes Atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$ die folgenden Schritte aus.
   (a) Falls
   $$K \cap \{A_1, \ldots, A_m\} \neq \emptyset \wedge K \cap \{B_1, \ldots, B_n\} \neq \emptyset,$$
   so lösche $a$ aus $\varphi$.
   (b) Falls
   $$K \cap \{A_1, \ldots, A_m\} \neq \emptyset \wedge K \cap \{B_1, \ldots, B_n\} = \emptyset$$
   oder falls
   $$K \cap \{A_1, \ldots, A_m\} = \emptyset \wedge K \cap \{B_1, \ldots, B_n\} \neq \emptyset,$$
   so stoppe ohne Rückgabewert.
3. Rate nichtdeterministisch eine Menge $L' \subseteq \mathrm{Var}_\varphi$[13] und setze $\alpha(X) = \{0\}$ für alle $X \in L'$. Definiere dann $L = L' \cup \{C \in \mathrm{Const}_\varphi \mid \mathrm{C} = \{0\}\}$.
4. Führe für jedes Atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$ die folgenden Schritte aus.
   (a) Falls
   $$L \cap \{A_1, \ldots, A_m\} \neq \emptyset \wedge L \cap \{B_1, \ldots, B_n\} \neq \emptyset,$$
   so lösche $a$ aus $\varphi$.

[12] Offenbar ist die Menge $\{C \in \mathrm{Const}_\varphi \mid \mathrm{C} = \emptyset\}$ eine Teilmenge von $\{\emptyset\}$. Wir verwenden die ausführlichere Schreibweise, um zu verdeutlichen, dass $K$ alle Variablen und Konstanten enthält, die von $\alpha$ auf die leere Menge abgebildet werden.

[13] Man beachte hier, dass jedes Atom mit einer Variable aus $K$ gelöscht wurde und somit $\mathrm{Var}_\varphi$ keine Elemente aus $K$ mehr enthält.

(b) Falls

$$L \cap \{A_1, \ldots, A_m\} \neq \emptyset \wedge L \cap \{B_1, \ldots, B_n\} = \emptyset$$

oder falls

$$L \cap \{A_1, \ldots, A_m\} = \emptyset \wedge L \cap \{B_1, \ldots, B_n\} \neq \emptyset,$$

so stoppe ohne Rückgabewert.

5. Rate nichtdeterministisch eine Menge $M' \subseteq \mathrm{Var}_\varphi$.[14]

   Setze $M = M' \cup \{C \in \mathrm{Const}_\varphi \mid |\mathrm{C}| = 1\}$.

   Ersetze jedes Atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$ durch

$$\Big( \prod_{1 \leq i \leq m, A_i \notin M} A_i = \prod_{1 \leq i \leq n, B_i \notin M} B_i \Big) \wedge \Big( \prod_{1 \leq i \leq m, A_i \in M} A_i = \prod_{1 \leq i \leq n, B_i \in M} B_i \Big).$$

   Rate für jede Konstante $C \in M$ die Primfaktorzerlegung (falls „falsch“ geraten wurde, stoppe ohne Rückgabewert). Seien $p_1, \ldots, p_k$ die Primzahlen, die $c$ für eine Konstante $\{c\} \in M$ teilen. Speichere für jede Konstante $\{c\} \in M$ lediglich den Vektor $(e_1, \ldots, e_k)$ mit $c = \prod_{i=1}^{k} p_i^{e_i}$. Wir werden auch die Werte $\alpha(X)$ für Variablen $X \in M$ auf diese Weise speichern.

6. Führe die folgenden Schritte aus.

   (a) Für jedes Atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$, für welches alle auftretenden Variablen in $M$ sind, wende die folgenden Regeln an.

      i. Falls $\alpha(A_i)$ für jedes $i$ definiert ist, so gibt es einen Vektor $(e_1, \ldots, e_k)$, sodass $\prod_{i=1}^{k} p_i^{e_i}$ die eindeutige Zahl in $\prod_{i=1}^{m} \alpha(A_i)$ ist. Der Vektor wird dabei als komponentenweise Summe der Vektoren für die $\alpha(A_i)$ errechnet. Verfahre für jedes $B_i$ mit nicht definiertem $\alpha(B_i)$ wie folgt: Rate nichtdeterministisch einen Vektor $(e'_1, \ldots, e'_k)$ mit $e'_j \leq e_j$. Speichere $\alpha(B_i)$ als $(e'_1, \ldots, e'_k)$.

      ii. Gehe analog vor, falls $\alpha(B_i)$ für jedes $i$ definiert ist.

   (b) Für jedes Atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$, für das alle auftretenden Variablen nicht in $M$ sind, wende die folgenden Regeln an.

      i. Falls alle $\alpha(A_i) = [a_i, a'_i]$ definiert sind, so rate nichtdeterministisch für jedes $B_i$ mit noch nicht definiertem $\alpha(B_i)$ zwei Zahlen $s_1 < s_2$ mit

$$s_2 \leq \max(\{a'_1, \ldots, a'_m\})$$

      und definiere $\alpha(B_i) =_{\mathrm{def}} [s_1, s_2]$.

      ii. Gehe, falls alle $\alpha(B_i)$ definiert sind, analog vor.

   (c) Falls in den Schritten 6a und 6b für eine Variable $X$ der Wert $\alpha(X)$ definiert wurde, so führe den Schritt 6 erneut aus.

7. Für jede noch nicht definierte Variable $X$ setze $\alpha(X) =_{\mathrm{def}} \emptyset$.[15]

[14] Die Variablen aus $M$ werden mit einelementigen Mengen belegt. Zudem gilt der analoge Hinweis wie in der letzten Fußnote.

[15] Dieser Schritt ist nicht nötig, da es einen Rechenweg gibt, auf dem die entsprechenden Variablen bereits in Schritt 1 auf die leere Menge gesetzt und anschließend entfernt wurden.

8. Prüfe für jedes Atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$, für das alle auftretenden Variablen in $M$ sind, ob es erfüllt ist. Dabei ist lediglich die jeweilige Summe der Vektoren von Exponenten zu bilden.

   Falls $a$ nicht erfüllt ist, stoppe ohne Rückgabewert. Andernfalls streiche $a$ aus $\varphi$. Bezeichne die so erzeugte Formel mit $\psi$.

9. Gib $(\psi, \alpha_{|\mathrm{var}_\psi})$ zurück.

Der Beweis ist vollständig, wenn die folgenden vier Aussagen gezeigt sind:

1. Falls auf einem Rechenweg eine Formel $\psi$ zurückgegeben wurde, so ist diese nicht in $\mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$, wenn $\varphi \notin \mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$ ist.
2. Falls $\varphi \in \mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$, so gibt $\mathcal{A}$ auf mindestens einem Rechenweg ein Paar $(\psi, \alpha)$ zurück, sodass $\alpha$ eine erfüllende Variablenbelegung für $\psi$ ist.
3. $\mathcal{A}$ arbeitet in nichtdeterministischer Polynomialzeit.
4. Für jedes zurückgegebene Paar $(\psi, \alpha)$ und jede Variable $X$ aus $\psi$ gilt $\max(\alpha(X)) \leq x$.

**1.)** Offenbar gilt: Wenn die Eingabeformel nicht in $\mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$ ist, so ist auch die nach Ausführung von Schritt 4 berechnete Formel nicht in $\mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$.

Für Intervalle $A_1, \ldots, A_m, B_1, \ldots, B_n$ gilt:

$$\prod_{1 \leq i \leq m, |A_i|=1} A_i = \prod_{1 \leq i \leq n, |B_i|=1} B_i \wedge \prod_{1 \leq i \leq m, |A_i| \neq 1} A_i = \prod_{1 \leq i \leq n, |B_i| \neq 1} B_i \Rightarrow \prod_{i=1}^{m} A_i = \prod_{i=1}^{n} B_i.$$

Hat also die Formel nach den Umformungen in Schritt 5 eine erfüllende Belegung, dann auch davor.

Da in Schritt 8 überprüft wird, ob alle Atome mit Variablen und Konstanten aus $M$ von der konstruierten Belegung erfüllt werden und die Variablen und Konstanten aus $M$ nicht in den verbleibenden Atomen auftreten, folgt die Behauptung.

**2.)** Sei $\varphi \in \mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$ und $\beta$ eine erfüllende Variablenbelegung für $\varphi$. Dann gibt es einen Rechenweg, auf dem $\mathcal{A}$ die Mengen $K, L, M$ so rät, dass für alle Variablen $X$ der Eingabeformel

$$\begin{aligned} X \in K &\Leftrightarrow \beta(X) = \emptyset \\ X \in L &\Leftrightarrow \beta(X) = \{0\} \\ X \in M &\Leftrightarrow \beta(X) \in \{\mathbb{N}\} - \{\{0\}\} \end{aligned}$$

gilt. Auf diesem Rechenweg terminiert $\mathcal{A}$ bis einschließlich Schritt 5 nicht und die nach Ausführung von Schritt 4 vorliegende Formel wird von $\beta$ erfüllt.

Weil für die verbleibenden Variablen $X$ die Menge $\beta(X)$ nicht leer und ungleich $\{0\}$ ist, erfüllt $\beta$ wegen Lemma 4.8 auch die Formel, die $\mathcal{A}$ nach Ausführung von Schritt 5 erzeugt hat.

Offenbar gibt es einen Rechenweg, auf dem für jede in Schritt 6a definierte Variable $X$

$$\alpha(X) = \beta(X)$$

gilt.
Wegen Lemma 4.9 gilt dies auch für jede in Schritt 6b definierte Variable.

Deshalb gilt bei Erreichen von Schritt 7, dass jedes Atom, in dem für alle Variablen $X$ ein Wert $\alpha(X)$ definiert ist, von $\alpha$ erfüllt wird.
Für alle anderen Atome $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$ gibt es mindestens eine Variable $A_i$ und mindestens eine Variable $B_j$, sodass $\alpha(A_i)$ und $\alpha(B_j)$ noch nicht definiert sind. Andernfalls hätten wegen 6a und 6b alle Variablen in $a$ einen definierten Funktionswert unter $\alpha$.
Durch das Setzen von $\alpha(A_i) = \emptyset$ und $\alpha(B_j) = \emptyset$ erfüllt $\alpha$ das Atom $a$. Folglich erfüllt $\alpha$ die gesamte Formel und insbesondere auch die zurückgegebene Formel $\psi$.

**3.)** Die Schritte 6a und 6b werden offenbar höchstens $|\mathrm{Var}_\varphi|$ mal ausgeführt, wobei $\varphi$ an dieser Stelle die Eingabeformel bezeichne. Die einzigen Schritte im Algorithmus, für welche es nicht offensichtlich ist, dass sie in nichtdeterministischer Polynomialzeit ausgeführt werden können, sind somit die Schritte 6a und 8.
Sei $e$ der größte Exponent, der in der Primfaktorzerlegung einer Zahl $c$ für eine Konstante $C = \{c\}$ auftaucht. Seien die Variablen $X_1, \ldots, X_r$ die Variablen aus $M$, sodass $\alpha(X_i)$ vor $\alpha(X_j)$ für $i < j$ definiert wird. Sei ferner $x_i$ das eindeutige Element in $\alpha(X_i)$.

Induktiv kann gezeigt werden, dass der größte Exponent in der Primfaktorzerlegung von $x_i$ nicht größer als $e \cdot 2^i$ ist.
Insbesondere können also die Mengen $\alpha(X)$ in polynomiellem Raum gespeichert werden. Offenbar kann daher auch im Schritt 8 in polynomieller Zeit überprüft werden, ob ein Atom von $\alpha$ erfüllt wird.

**4.)** Diese Bedingung gilt wegen Schritt 6b und weil die Atome mit Variablen aus $M$ gelöscht wurden. □

Um zu entscheiden, ob $\varphi \in \mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$ gilt, ist also nur noch für ein von $\mathcal{A}$ zurückgegebenes Paar $(\psi, \alpha)$ zu überprüfen, ob $\alpha$ eine erfüllende Belegung für $\psi$ ist.
Wir definieren dazu das folgende Problem.

**Definition 4.11**
Sei IntervalEquality die Menge

$$\{([a_1, a_1'], \ldots, [a_m, a_m'], [b_1, b_1'], \ldots, [b_n, b_n']) \mid \forall i, j \ a_i < a_i', b_i < b_i' \wedge \prod_{i=1}^{m} [a_i, a_i'] = \prod_{i=1}^{n} [b_i, b_i']\}.$$

**Lemma 4.12**
*Es gilt* IntervalEquality $\in \Pi_2^{\mathrm{p}}$.

*Beweis.* Seien $A_1, \ldots, A_m, B_1, \ldots, B_n$ nichtleere Intervalle mit $|A_i|, |B_j| \geq 2$ für $i = 1, \ldots, m$ und $j = 1, \ldots, n$.

Es gilt

$$\prod_{i=1}^{m} A_i = \prod_{i=1}^{n} B_i \Leftrightarrow \forall x_1 \in A_1 \ldots \forall x_m \in A_m \forall y_1 \in B_1 \ldots \forall y_n \in B_n$$
$$\exists x'_1 \in B_1 \ldots \exists x'_n \in B_n \exists y'_1 \in A_1 \ldots \exists y'_m \in A_m$$
$$\prod_{i=1}^{m} x_i = \prod_{i=1}^{n} x'_i \wedge \prod_{i=1}^{n} y_i = \prod_{i=1}^{m} y'_i.$$

Da die $x_i, y_i, x'_i, y'_i$ polynomielle Länge haben, gilt $A \in \forall^{\mathrm{P}}\exists^{\mathrm{P}}\mathrm{P} = \Pi_2^{\mathrm{p}}$. □

Damit lässt sich für $\mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$ die folgende obere Schranke zeigen.

**Satz 4.13**
$\mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big) \in \Sigma_3^{\mathrm{p}}$.

*Beweis.* Gemäß Satz 4.10 und Lemma 4.12 kann $\mathrm{CSP}\Big([\mathbb{N}], \{=, \times\}\Big)$ von einem NP-Algorithmus mit $\Pi_2^{\mathrm{p}}$-Orakel entschieden werden. Es folgt die Behauptung. □

**Bemerkung 4.14**
*Unser Entscheidungsalgorithmus für* IntervalEquality *überprüft die Gleichheit der beiden Produkte von Intervallen, indem er alle Elemente der beiden Produkte betrachtet. Möglicherweise geht dies effizienter. Es wurde bereits in Lemma 4.9 gezeigt, dass für eine Gleichung von Produkten von Intervallen die maximale obere Intervallgrenze bei beiden Produkten übereinstimmt. Vielleicht lässt sich sogar ein allgemeineres Resultat zeigen, sodass ein Algorithmus lediglich die Intervallgrenzen der einzelnen Intervalle betrachten muss, um* IntervalEquality *zu entscheiden. Ein solcher Algorithmus könnte in nichtdeterministischer Polynomialzeit arbeiten.*

## $\mathrm{CSP}\Big(\mathcal{P}_{\mathbf{fin}}(\mathbb{N}), \{=, \times\}\Big)$

Wir verwenden eine abkürzende Schreibweise für die Menge der Primteiler von in einer Formel auftretenden Zahlen:

**Definition 4.15**
Es sei $\varphi$ eine $\{\times\}$-Formel. Wir definieren

$$P_\varphi = \{p \mid \ p \text{ prim und } p \mid c \text{ für ein } 0 \neq c \in C, C \in \mathrm{Const}_\varphi\}.$$

Unmittelbar aus der Definition ergibt sich die Aussage:

**Lemma 4.16**
*Es sei $\varphi$ eine $\{\times\}$-Formel und $n =_{def} |\varphi|$. Dann gilt $|P_\varphi| \in O(n^2)$.*[16]

*Beweis.* Es enthält $P_\varphi$ die Primfaktoren von in Konstanten von $\varphi$ auftretenden Zahlen. Die Anzahl dieser Zahlen ist offenbar nicht größer als $n$ und jede dieser Zahlen $\alpha$ hat höchstens die Länge $n$ und ist damit $\in O(2^n)$. Somit hat $\alpha$ höchstens $n$ Primteiler und es gilt $|P_\varphi| \in O(n^2)$. □

[16] Offenbar kann man hier auch eine bessere obere Schranke beweisen, nämlich $O(n)$. Die im Folgenden erwähnte Schranke ist jedoch für unsere Zwecke hinreichend.

Das folgende Lemma schränkt den Raum, in dem wir nach einer erfüllenden Belegung zu suchen haben, stark ein.
So brauchen wir lediglich Variablenbelegungen zu betrachten, in deren Wertebereich ausschließlich Mengen von Zahlen sind, deren Primteiler auch Primteiler eines Elements einer Konstanten sind.

**Lemma 4.17**
*Es sei* $\varphi \in \mathrm{CSP}'\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times\}\Big)$ *und* $\alpha$ *eine erfüllende Variablenbelegung für* $\varphi$*. Dann ist auch* $\beta$ *mit* $\beta(X) = \emptyset$*, falls* $\alpha(X) = \emptyset$ *und*

$$\begin{aligned}\beta(X) &= \Big\{y \mid \exists x \in \alpha(X) : \; y = \frac{x}{\prod_{p \in \mathbb{P} - P_\varphi,\, p^e \mid x,\, p^{e+1} \nmid x} p^e}\Big\} \\ &= \{y \mid \exists x \in \alpha(X) : \; y \textit{ entsteht aus } x \textit{ durch Streichen aller Primfaktoren } p \notin P_\varphi\},\end{aligned}$$

*falls* $\alpha(X) \neq \emptyset$ *gilt, eine erfüllende Variablenbelegung.*

*Beweis.* Sei $A \times B = C$ ein beliebiges Atom aus $\varphi$. Dann gilt $\alpha(A) \times \alpha(B) = \alpha(C)$. Falls $\emptyset \in \{\alpha(A), \alpha(B), \alpha(C)\}$, so erfüllt offenbar auch $\beta$ das Atom. Andernfalls folgt

$$\begin{aligned}\beta(C) &= \Big\{y \mid \exists x \in \alpha(C) : \; y = \frac{x}{\prod_{p \in \mathbb{P} - P_\varphi,\, p^e \mid x,\, p^{e+1} \nmid x} p^e}\Big\} \\ &= \Big\{y \mid \exists x \in \alpha(A) \times \alpha(B) : \; y = \frac{x}{\prod_{p \in \mathbb{P} - P_\varphi,\, p^e \mid x,\, p^{e+1} \nmid x} p^e}\Big\} \\ &= \Big\{y \mid \exists x \in \alpha(A) \; y = \frac{x}{\prod_{p \in \mathbb{P} - P_\varphi,\, p^e \mid x,\, p^{e+1} \nmid x} p^e}\Big\} \times \\ &\quad \Big\{y \mid \exists x \in \alpha(B) : \; y = \frac{x}{\prod_{p \in \mathbb{P} - P_\varphi,\, p^e \mid x,\, p^{e+1} \nmid x} p^e}\Big\} \\ &= \beta(A) \times \beta(B).\end{aligned}$$

□

Wir können unseren Suchraum noch weiter einschränken.

**Lemma 4.18**
*Sei* $\varphi \in \mathrm{CSP}'\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times\}\Big)$ *mit* $n =_{def} |\varphi|$*. Definiere*

$$\zeta = \max(\{e \mid \; p^e \mid c \textit{ für } p \in P_\varphi \textit{ und } c \in C \textit{ für ein } C \in \mathrm{Const}_\varphi \;\}).$$

*Dann existiert eine erfüllende Variablenbelegung* $\alpha$ *für* $\varphi$ *mit*

$$\forall X \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi : \; \alpha(\mathrm{X}) \subseteq \Big\{ \prod_{\mathrm{p_i} \in \mathrm{P}_\varphi} \mathrm{p_i^{e_i}} \mid \forall \mathrm{i} : \; \mathrm{e_i} \leq \zeta \cdot 2^{\mathrm{n}} \Big\} \cup \{0\}.$$

*Beweis.* Wir geben einen nichtdeterministischen Algorithmus an, der eine erfüllende Variablenbelegung zurückgibt, die den oben formulierten Anforderungen genügt. Der Algorithmus konstruiert dazu sukzessive eine totale Abbildung $\alpha : \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi \to \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$, aus der wir am Ende die genannte Variablenbelegung erhalten.

1. Setze $\alpha(C) = C$ für alle Konstanten $C$.
2. Rate nichtdeterministisch eine Teilmenge $K'$ von $\mathrm{Var}_\varphi$ und setze $\alpha(X) = \emptyset$ für alle $X \in K'$. Setze $K = K' \cup \{C \in \mathrm{Const}_\varphi \mid \mathrm{C} = \emptyset\}$[17].
3. Lösche aus $\varphi$ jedes Atom $A \times B = C$ mit $A, C \in K$ oder $B, C \in K$.

   Falls für ein Atom $A \times B = C$

   $$(C \in K \ \wedge \ A, B \notin K) \ \vee \ (\{A, B\} \cap K \neq \emptyset \ \wedge \ C \notin K)$$

   gilt, so stoppe die Berechnung ohne Rückgabewert.
4. Rate nichtdeterministisch eine Teilmenge $L'$ von $\mathrm{Var}_\varphi - \mathrm{K}$ und setze $\alpha(X) = \{0\}$ für alle $X \in L'$. Setze $L = L' \cup \{C \in \mathrm{Const}_\varphi \mid \mathrm{C} = \{0\}\}$[17].
5. Lösche aus $\varphi$ jedes Atom $A \times B = C$ mit $A, C \in L$ oder $B, C \in L$.

   Falls für ein Atom $A \times B = C$ gilt: $(C \in L \wedge A, B \notin L)$ oder $(\{A, B\} \cap L \neq \emptyset \wedge C \notin L)$, so stoppe die Berechnung ohne Rückgabewert.
6. Führe folgende Schritte aus:
   (a) Für jedes Atom $A \times B = C$, für das $\alpha(C)$ definiert[18] und $\alpha(A)$ nicht definiert ist, rate nichtdeterministisch eine Teilmenge $S \neq \emptyset$ und $S \neq \{0\}$ von

   $$\Big\{ \prod_{p_i \in P_\varphi} p_i^{e_i} \mid \exists x \in \alpha(C) - \{0\} \forall i : \ p_i^{e_i} \mid x \Big\} \cup \{0\}$$

   und setze $\alpha(A) = S$. Verfahre für $B$ analog.
   (b) Für jedes Atom $A \times B = C$, für das $\alpha(A)$ und $\alpha(B)$ definiert und $\alpha(C)$ nicht definiert ist, setze $\alpha(C) = \alpha(A) \times \alpha(B)$.
   (c) Falls in 6a oder 6b ein Wert $\alpha(X)$ für eine Variable $X$ definiert wurde, führe Schritt 6 erneut aus.
7. Für alle $X \in \mathrm{Var}_\varphi$, für die $\alpha(X)$ noch nicht definiert ist, setze $\alpha(X) = \emptyset$[19].
8. Setze $\alpha =_{\mathrm{def}} \alpha_{|\mathrm{Var}_\varphi}$. Falls $\alpha$ total und erfüllend ist, gib $\alpha$ zurück. Stoppe andernfalls ohne Rückgabewert.

Wir zeigen, dass der Algorithmus auf mindestens einem Rechenweg eine erfüllende Belegung zurückgibt:

Sei $\beta$ eine erfüllende Variablenbelegung für $\varphi$, sodass für jede Variable $X$ die Menge $\beta(X)$ nur Zahlen enthält, deren Primteiler in $P_\varphi$ sind. Eine solche Belegung existiert nach Lemma

[17] Für den weiteren Beweis wäre es auch in Ordnung, $K = K' \cup \{\emptyset\}$ bzw. $L = L' \cup \{\{0\}\}$ zu setzen. Falls $\emptyset \notin \mathrm{Const}_\varphi$ würden sich so die beiden Definition zwar unterscheiden, für den weiteren Beweis wäre dies jedoch nicht von Relevanz. Dennoch schreiben wir es ausführlicher, damit deutlicher wird, dass wir in $K$ alle Variablen und Konstanten haben wollen, die mit der leeren Menge belegt werden bzw. die leer sind. Wir werden in den folgenden Schritten analog vorgehen.

[18] Man beachte im Folgenden, dass $\alpha(C) \notin \{\emptyset, \{0\}\}$ gilt

[19] Dieser Schritt ist an sich überflüssig, weil es für jede mögliche Auswahl an Variablen, die mit $\emptyset$ zu belegen sind, bereits einen entsprechenden Rechenweg gibt. Dennoch erleichtert dieser Schritt des Algorithmus die Argumentation an späterer Stelle.

4.17. Dann gibt es einen Rechenweg, auf dem der obige Algorithmus die Mengen $K$ und $L$ so wählt, dass für alle Variablen $X$

$$X \in K \Leftrightarrow \beta(X) = \emptyset$$
$$X \in L \Leftrightarrow \beta(X) = \{0\}$$

gilt. Wir bezeichnen im Folgenden mit $\alpha$ die Belegung, die auf dem von uns angedeuteten, aber noch nicht bis zum Ende skizzierten Rechenweg konstruiert wird. Es stimmen $\alpha$ und $\beta$ auf allen bis einschließlich Schritt 5 belegten Variablen überein. Die in den Schritten 1 bis 5 gelöschten Atome sind somit unabhängig von der weiteren Definition von $\alpha$ unter jeder Termbelegung, die auf den bereits definierten Variablen mit $\alpha$ übereinstimmt, erfüllt und müssen nicht weiter betrachtet werden.

In Schritt 6 wird lediglich der Wert $\alpha(X)$ für solche Variablen $X$ definiert, die ausschließlich in Atomen $A \times B = C$ vorkommen, für die $\beta(A), \beta(B), \beta(C) \notin \{\emptyset, \{0\}\}$ gilt[20]. Induktiv lässt sich daher zeigen, dass es einen Rechenweg gibt, auf welchem auch nach dem Schritt 6 die Belegungen $\alpha$ und $\beta$ auf allen bislang belegten Variablen übereinstimmen.
Insbesondere gilt also für jedes Atom $A \times B = C$, für welches die Werte $\alpha(A)$, $\alpha(B)$ und $\alpha(C)$ nach Schritt 6 definert sind:

$$\alpha(A) \times \alpha(B) = \alpha(C).$$

Sei $A \times B = C$ ein Atom, sodass für mindestens eine der Variablen nach Schritt 6 noch kein Wert unter $\alpha$ definiert ist. Dann sind $\alpha(C)$ und mindestens einer der zwei Werte $\alpha(A)$ und $\alpha(B)$ noch nicht definiert: Angenommen, dies wäre nicht so, dann wäre

- $\alpha(C)$ definiert, wodurch aber nach Schritt 6a auch $\alpha(A)$ und $\alpha(B)$ definiert wären, oder es wären
- $\alpha(A)$ und $\alpha(B)$ bereits definiert, wodurch aber nach Schritt 6b auch $\alpha(C)$ definiert wäre.

Wir erhalten also jeweils einen Widerspruch.
Somit ist $\alpha$ erfüllend, wenn wir für jede nach Schritt 6 noch nicht belegte Variable $X$

$$\alpha(X) = \emptyset$$

setzen. Dies geschieht in Schritt 7.

Es bleibt

$$\forall X \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi : \ \alpha(\mathrm{X}) \subseteq \Big\{ \prod_{\mathrm{p_i} \in \mathrm{P}_\varphi} \mathrm{p_i^{e_i}} \mid \forall \mathrm{i} : \ 0 \leq \mathrm{e_i} \leq \zeta \cdot 2^{\mathrm{n}} \Big\}$$

zu zeigen. Es seien die Variablen $X_1, \ldots, X_{|\mathrm{Var}_\varphi|}$ so indiziert, dass für $i < j$ der Wert $\alpha(X_i)$ vor dem Wert $\alpha(X_j)$ definiert wird. Induktiv kann man leicht zeigen, dass für alle $j = 1, \ldots, |\mathrm{Var}_\varphi|$

$$\alpha(X_j) \subseteq \Big\{ \prod_{p_i \in P} p_i^{e_i} \mid \forall i : \ e_i \leq \zeta \cdot 2^j \Big\}$$

gilt. Wegen $|\mathrm{Var}_\varphi| \leq \mathrm{n}$ folgt die Behauptung.

□

[20] Man beachte hier, dass wir einen Teil der Atome der Eingabeformel gelöscht haben.

Das vorausgehende Lemma zeigt, dass es genügt, eine endliche Menge von Variablenbelegungen im Hinblick darauf zu testen, ob sie erfüllend sind, um für eine gegebene $\{\times\}$-Formel zu entscheiden, ob sie wahr ist. Damit folgt direkt die Entscheidbarkeit von $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times\}\Big)$.

Das Lemma macht aber auch eine Aussage darüber, wie groß die zu testende Menge von Variablenbelegungen ist. Damit lassen sich Aussagen über die Zugehörigkeit zu einer konkreten Komplexitätsklasse machen.

Dabei ist der folgende Aspekt wesentlich: Obwohl es sein kann, dass für jede erfüllende Belegung $\alpha$ eine Variable $X$ mit exponentiell langem $\max(\alpha(X))$ existiert, hat nach dem eben bewiesenen Lemma $\alpha(X)$ für alle Variablen $X$ höchstens exponentiell viele Elemente.

**Satz 4.19**
*Es gilt* $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times\}\Big) \in \mathrm{NEXP}$.

*Beweis.* Gemäß Lemma 4.18 entscheidet der im Folgenden beschriebene Algorithmus das Problem $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times\}\Big)$.

- Eingabe: $\{\times\}$-Formel $\varphi'$.
- Berechne $\varphi = f(\varphi')$ für die FL-Funktion $f$, die $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times\}\Big)$ auf das Problem $\mathrm{CSP}'\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times\}\Big)$ reduziert. Es hat $\varphi$ folglich ausschließlich Atome der Form $A \times B = C$.
- Bestimme $n = |\varphi|$ und

$$\zeta = \max(\{e \mid \exists C \in \mathrm{Const}_\varphi \, \exists \mathrm{p} \in \mathbb{P} \, \exists \mathrm{c} \in \mathrm{C} : \ \mathrm{p}^{\mathrm{e}} \mid \mathrm{c}\}).$$

- Berechne die Menge $P_\varphi = \{p_1, \ldots, p_{|P_\varphi|}\}$.
- Rate für jede Variable $X$ von $\varphi$ eine Menge

$$S \subseteq \Big\{ \prod_{i=1}^{|P_\varphi|} p_i^{e_i} \mid \forall i : \ 0 \leq e_i \leq \zeta \cdot 2^n \Big\} \cup \{0\}$$

  und setze $\alpha(X) = S$. Setze ferner $\alpha(C) = C$ für alle $C \in \mathrm{Const}_\varphi$.
- Falls es ein Atom $A \times B = C$ in $\varphi$ mit $\alpha(A) \times \alpha(B) \neq \alpha(C)$ gibt, gib 0 zurück. Andernfalls gib 1 zurück.

Der Algorithmus kann in $2^{p(n)}$ Rechenschritten für ein geeignetes Polynom $p$ berechnet werden, da lediglich $|\mathrm{Var}_\varphi| \in \mathrm{O(n)}$ Teilmengen $S$ von

$$\Big\{ \prod_{p_i \in P_\varphi} p_i^{e_i} \mid \forall i : \ e_i \leq \zeta \cdot 2^n \Big\} \cup \{0\}$$

für $\zeta \in O(2^n)$ zu raten und die einzelnen Atome zu überprüfen sind. Man beachte dabei, dass $S$ zwar exponentiell lange Zahlen enthalten kann, jedoch nur exponentiell viele Zahlen enthält. □

### 4.2.2 Untere Schranken

Für $M \in \{[\mathbb{N}], \mathcal{P}_{\text{fin}}(\mathbb{N})\}$ beweisen wir die $\leq_{\text{m}}^{\log}$-Härte von $\text{CSP}\Big(\text{M}, \{=, \times\}\Big)$ für NP. Für $M = \mathcal{P}_{\text{fin}}(\mathbb{N})$ könnte diese Aussage auch über eine Reduktion von $\text{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +\}\Big)$ gezeigt werden, da bei der Reduktion MSOS $\leq_{\text{m}}^{\log}$ $\text{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +\}\Big)$ eine Formel erzeugt wurde, deren Konstanten einelementig oder leer sind. Dies wird an späterer Stelle aus dem Beweis von Lemma 4.24 hervorgehen. Da jedoch $[\mathbb{N}]$ bezüglich der Multiplikation nicht abgeschlossen ist, kann im Fall $M = [\mathbb{N}]$ nicht auf diese Weise vorgegangen werden. Wir zeigen deshalb für beide Fälle parallel die Aussage MSOS $\leq_{\text{m}}^{\log}$ $\text{CSP}\Big(\text{M}, \{=, \times\}\Big)$ und damit:

**Satz 4.20**
$\text{CSP}\Big(\text{M}, \{=, \times\}\Big)$ *ist* $\leq_{\text{m}}^{\log}$*-hart für* NP.

*Beweis.* Wir gehen analog zum Beweis von Satz 4.7 vor.
Seien $a_1, \ldots, a_k, b \in \mathbb{N}$ gegeben. Wir konstruieren eine $\{\times\}$-Formel $\varphi$, sodass die Bedingung $\varphi \in \text{CSP}\Big(\text{M}, \{=, \times\}\Big)$ genau dann gilt, wenn $x_1, \ldots, x_k \in \mathbb{N}$ mit $\sum_{i=1}^{k} x_i a_i = b$ existieren. Es gilt

$$\sum_{i=1}^{k} x_i a_i = b \Leftrightarrow \prod_{i=1}^{k} \left(2^{x_i} 2^{a_i}\right) = 2^b.$$

Wir verwenden zur kurzen Darstellung der Zahlen $2^s$ für $s \in \mathbb{N}$ in $\varphi$ die Technik Square-And-Multiply. Sei dazu $b_m \ldots b_0$ die Binärdarstellung von $s$. Dann gilt

$$2^s = 2^{\sum_{i=0}^{m} b_i 2^i} = \prod_{0 \leq i \leq m} \left(2^{2^i}\right)^{b_i} = \prod_{0 \leq i \leq m, b_i = 1} 2^{2^i}.$$

Es sei im Folgenden $b_{i,m_i} \ldots b_{i,0}$ die Binärdarstellung von $a_i$, sowie $b_{k+1,m_{k+1}} \ldots b_{k+1,0}$ die Binärdarstellung von $b$.

Wir konstruieren für $i \in \{1, \ldots, k+1\}$ eine Formel $\psi_i$, welche eine Variable $A_i$ enthält, die unter jeder erfüllenden Belegung den Wert $2^{a_i}$ für $i \in \{1, \ldots, k\}$ und den Wert $2^b$ für $i = k+1$ zugewiesen bekommt. Sei dazu $m = \max(m_1, \ldots, m_{k+1})$.
Es sei fener

$$\chi = \exists Y_m \ldots \exists Y_0 \ \big(Y_0 = \{2\}\big) \wedge \bigwedge_{j=0}^{m-1} \big(Y_{j+1} = Y_j \times Y_j\big)$$

und

$$\psi_i = \exists A_i \big(A_i = \prod_{0 \leq j \leq m_i, b_j = 1} Y_j\big).$$

Sei $\varphi$ die Formel, die aus

$$\exists X_1 \ldots \exists X_k \ \Big(\chi \wedge \bigwedge_{i=1}^{k+1} \psi_i \wedge A_{k+1} = \prod_{i=1}^{k} \big(A_i \times X_i\big)\Big)$$

entsteht, wenn alle existentiellen Quantifizierungen nach vorne gezogen werden.

Sei $\alpha$ eine erfüllende Variablenbelegung von $\varphi$. Offenbar weist $\alpha$ jeder Variable $Y_j$ und jeder Variable $A_i$ eine einelementige Menge zu. Gäbe es eine Variable $X_i$, der von $\alpha$ eine

nicht einelementige Menge zugewiesen wird, so wäre das Atom $A_{k+1} = \prod_{i=1}^{k}(A_i \times X_i)$ nicht erfüllt. Also gilt $W_\alpha \subseteq \{\mathbb{N}\}$.

Es genügt daher zu zeigen, dass genau dann $x_1, \ldots, x_n \in \mathbb{N}$ mit $\prod_{i=1}^{k} 2^{x_i} 2^{a_i} = 2^b$ existieren, wenn es für $\varphi$ eine erfüllende Belegung $\alpha$ mit $W_\alpha \subseteq \{\mathbb{N}\}$ gibt.

„$\Rightarrow$“: Seien $x_1, \ldots, x_n \in \mathbb{N}$, sodass $\prod_{i=1}^{k} 2^{x_i} 2^{a_i} = 2^b$ gilt. Dann erfüllt offenbar jede Termbelegung mit $X_i \mapsto \{2^{x_i}\}$ die Formel $\varphi$.

„$\Leftarrow$“: Unter jeder erfüllenden Termbelegung $\alpha$ gilt $\alpha(A_i) = \{2^{a_i}\}$ für $i = 1, \ldots, k$ und $\alpha(A_{k+1}) = \{2^b\}$. Wenn wir das eindeutige Element in $\alpha(X_i)$ mit $x_i'$ bezeichnen, so gilt $\prod_{i=1}^{k} x_i' 2^{a_i} = 2^b$. Insbesondere sind also die $x_i'$ Zweierpotenzen und für $x_1, \ldots, x_n$ mit $x_i = \log_2 x_i'$ gilt $\prod_{i=1}^{k} 2^{x_i} 2^{a_i} = 2^b$.

Da die in der Eingabe auftretenden Zahlen bereits in Binärdarstellung vorliegen, kann die Formel $\varphi$ in logarithmischem Raum berechnet werden. □

## 4.3 $\mathbf{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, \cap\})$

Im Gegensatz zu der Variante über beliebigen endlichen Teilmengen von $\mathbb{N}$, die wir im Abschnitt 4.5 noch kurz diskutieren werden, lässt sich hier relativ leicht eine obere Schranke finden: Wir zeigen unter Verwendung von Integer Linear Programs die Zugehörigkeit des Problems $\text{CSP}\Big([\mathbb{N}], \{=, +, \cap\}\Big)$ zu NP.

Ferner ergibt sich die NP-Härte aus der NP-Härte von $\text{CSP}\Big([\mathbb{N}], \{=, +\}\Big)$.

Wir definieren eine bekanntlich in NP liegende Variante von Integer Linear Programs:

**Definition 4.21**
Es sei

$$\text{ILP} = \Big\{(A, b) \mid A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}^m,\, m, n \in \mathbb{N}^+, \exists x \in \mathbb{N}^n :\ Ax \leq b\Big\},$$

wobei $(d_1, \ldots, d_m) \leq (e_1, \ldots, e_m)$ genau dann gelte, wenn für alle $i = 1, \ldots, m$ die Bedingung $d_i \leq e_i$ erfüllt ist.

**Satz 4.22**
*Es gilt* $\text{CSP}\Big([\mathbb{N}], \{=, +, \cap\}\Big) \in \text{NP}$.

*Beweis.* Wir beweisen die Aussage durch Angabe eines nichtdeterministischen Polynomialzeitalgorithmus. Dabei genügt es gemäß Lemma 2.5, $\text{CSP}'\Big([\mathbb{N}], \{=, +, \cap\}\Big) \in \text{NP}$ zu zeigen.

Sei also $\varphi$ eine $\{+, \cap\}$-Formel, deren Konstanten Intervalle sind. Das heißt, dass jedes Atom die Form $X \oplus Y = Z$ für $X, Y, Z \in \text{Var}_\varphi \cup \text{Const}_\varphi$ und $\oplus \in \{+, \cap\}$ hat.

Rate nichtdeterministische eine Teilmenge $M$ von Variablen in $\varphi$ und ersetze jedes Auftreten einer Variable in $M$ durch $\emptyset$. Lösche jedes Atom $X \oplus Y = Z$ für $\oplus \in \{\cap, +\}$, falls $Z = \emptyset$ und $(X = \emptyset \vee Y = \emptyset)$. Bleibt kein Atom übrig, so gib $\{0\} + \{0\} = \{0\}$ zurück. Falls für ein Atom $X \oplus Y = Z$ for $\oplus \in \{\cap, +\}$ die Bedingung

$$\big((X = \emptyset \vee Y = \emptyset) \wedge Z \neq \emptyset\big) \vee \big(Z = \emptyset \wedge X, Y \neq \emptyset \wedge \oplus = +\big)$$

gilt, dann gib $\{0\} + \{0\} = \{1\}$ zurück. In allen anderen Fällen gib die – eventuell modifizierte – Formel zurück.

Damit gilt $\varphi \in \text{CSP}'\Big([\mathbb{N}], \{=, +, \cap\}\Big)$ genau dann, wenn auf mindestens einem Rechenweg eine Formel $\varphi'$ zurückgegeben wurde, die den folgenden Bedingungen genügt:

- Es gibt eine erfüllende Variablenbelegung mit Bildbereich $\subseteq [\mathbb{N}] - \{\emptyset\}$ für $\varphi'$ und,
- wenn es in $\varphi$ ein Atom $X \oplus Y = Z$ gibt, welches die leere Menge als Konstante enthält, so gilt $\oplus = \cap$, $Z = \emptyset$ und $X, Y \neq \emptyset$.

Wir geben einen Algorithmus an, der diese Bedingungen testet.

Wir definieren eine ILP-Instanz durch Angabe einer Reihe von Ungleichungen. Dadurch werden $A, b$ implizit festgelegt. Führe zwecks dessen für jedes $R \in (\text{Var}_\varphi \cup \text{Const}_\varphi) - \{\emptyset\}$ zwei ILP-Variablen $r_0, r_1$ ein. Setze für jede nichtleere Konstante $R = [l, u]$ weiterhin $r_0 = l$ und $r_1 = u$.

1. Für jedes Atom $X+Y=Z$ stellen wir die Gleichungen $x_0+y_0=z_0$ und $x_1+y_1=z_1$ auf.

2. Für jedes Atom $X \cap Y = Z$ mit $Z \neq \emptyset$ führe vier weitere Variablen $d, e, d', e'$ ein. Letztlich soll $z_0 = \max(x_0, y_0)$ und $z_1 = \min(x_1, y_1)$ ausgedrückt werden.

   - Zu $z_0 = \max(x_0, y_0)$: Dies kann durch $x_0 \leq z_0$, $y_0 \leq z_0$, $z_0 = dx_0 + ey_0$ und $d + e = 1$ ausgedrückt werden.
   - Zu $z_1 = \min(x_1, y_1)$: Dies kann durch $x_1 \geq z_1$, $y_1 \geq z_1$, $z_1 = d'x_1 + e'y_1$ und $d' + e' = 1$ ausgedrückt werden.

3. Für jedes Atom $X \cap Y = Z$ mit $Z = \emptyset$ möchten wir $y_1 < x_0 \vee x_1 < y_0$ fordern. Daher raten wir ein Bit $b$. Wenn $b = 0$, fügen wir die Ungleichung $y_1 < x_0$ hinzu. Andernfalls wird $x_1 < y_0$ hinzugefügt.

4. Führe weiterhin für jedes Paar von ILP-Variablen $x_0, x_1$, welche die untere und obere Grenze eines Intervalls beschreiben, die Ungleichung $x_0 \leq x_1$ hinzu.

Nach Konstruktion gilt: Genau dann wird auf mindestens einem Rechenweg eine ILP-Instanz $(A, b)$ mit $(A, b) \in \text{ILP}$ zurückgegeben, wenn die Formel $\varphi'$ eine erfüllende Variablenbelegung besitzt, von der keine Variable auf die leere Menge abgebildet wird.
Somit entscheidet der Algorithmus $\text{CSP}\Big([\mathbb{N}], \{=, +, \cap\}\Big)$.

Wegen $\text{ILP} \in \text{NP}$ arbeitet der Algorithmus in nichtdeterministischer Polynomialzeit. □

**Korollar 4.23**
$\text{CSP}\Big([\mathbb{N}], \{=, +, \cap\}\Big)$ *ist* $\leq_{\text{m}}^{\log}$*-vollständig für* NP.

*Beweis.* $\text{CSP}\Big([\mathbb{N}], \{=, +, \cap\}\Big) \in \text{NP}$ folgt aus Satz 4.22. Die Härte ergibt sich aus Satz 4.7 und Lemma 2.3. □

## 4.4 Untere Schranken für CSPs mit einer arithmetischen Operation und Mengenoperationen

In diesem Abschnitt beweisen wir verschiedene untere Schranken für CSPs über $\mathcal{P}_{\text{fin}}(\mathbb{N})$, die neben genau einer arithmetischen auch mindestens eine Mengenoperation enthalten. Genauer werden wir die $\Pi_2^{\text{p}}$-Härte für

- $\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, \cup\}\big)$
- $\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, \times, \cup\}\big)$

und die PSPACE-Härte für

- $\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, \oplus\}\big)$ für $\oplus \in \{\cap, -\}$
- $\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, \times, \oplus\}\big)$ für $\oplus \in \{\cap, -\}$

zeigen.

Die beiden Schranken folgen beinahe direkt aus der Literatur und sollten zumindest in manchen Fällen verbessert werden können. Unser Hauptaugenmerk war jedoch auf andere Fragen, insbesondere auf die Frage nach der Entscheidbarkeit der besagten Probleme, ausgerichtet. Die genaue Betrachtung der unteren Schranken steht damit noch aus.

Um uns bei dem Beweis der unteren Schranken in einigen Fällen auf die Variante mit der Addition beschränken zu können, formulieren und beweisen wir das folgende Lemma.

**Lemma 4.24**
*Es gilt für $O \subseteq \{\cap, \cup, -\}$ mit $O \cap \{\cup, -\} \neq \emptyset$*

$$\text{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +\} \cup \text{O}) \leq_{\text{m}}^{\log} \text{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, \times\} \cup \text{O}).$$

*Beweis.* Sei $\varphi$ gegeben. Ersetze in $\varphi$ jedes $+$ durch $\times$. Ersetze zudem jede Konstante

$$C = \{k_1, \ldots, k_r\}$$

wie folgt:
Für $i \in \{1, \ldots, r\}$ sei $b_l \ldots b_0$ die Binärdarstellung von $k_i$. Es gilt dann

$$2^{k_i} = 2^{\sum_{j=0}^{l} b_j 2^j} = \prod_{j=0}^{l} \left(2^{2^j}\right)^{b_j} = \prod_{j \in [0,l], b_j = 1} 2^{2^j},$$

wobei

$$2^{2^j} = \left(2^{2^{j-1}}\right)^2$$

gilt.
Es lässt sich daher $\{2^{k_i}\}$ wie folgt beschreiben

$$\begin{aligned}\alpha_i = &\exists X_i \exists Y_0 \ldots \exists Y_l \ \big(Y_0 = \{2\}\big) \wedge \big(Y_1 = Y_0 \times Y_0\big) \wedge \big(Y_2 = Y_1 \times Y_1\big) \wedge \cdots \wedge \\ &\big(Y_l = Y_{l-1} \times Y_{l-1}\big) \wedge \Big(X_i = \prod_{j \in [0,l], b_j = 1} Y_j\Big).\end{aligned}$$

Für jede $\alpha_i$ erfüllende Belegung $\gamma$ gilt also $\gamma(X_i) = \{2^{k_i}\}$. Wir benennen nun in $\alpha_i$ die Variablen $Y_j$ in $Y_{i,j}$ um[21]. Betrachte die Formel die aus

$$\exists X_C \bigwedge_{1 \leq i \leq r} \alpha_i \wedge \Big( X_C = \bigcup_{1 \leq i \leq r} X_i \Big).$$

durch das Ziehen der existentiellen Quantifizierungen nach vorne entsteht und nenne sie $\psi_C$.
Unter einer $\psi_C$ erfüllenden Belegung $\gamma$ gilt $\gamma(X_C) = \{2^c \mid c \in C\}$.
Ersetze nun alle Auftreten von $C$ in $\varphi$ durch $X$, bilde die Konjunktion aus dieser Formel und $\bigwedge_{C \in \mathrm{Const}_\varphi} \psi_C$, ziehe anschließend alle existentiellen Quantoren an den Anfang der Formel und nenne die Formel $\varphi'$.

Wegen

$$\forall x, y, z \in \mathbb{N} \quad \Big( x + y = z \Leftrightarrow 2^x \cdot 2^y = 2^z \Big)$$

gilt dann, dass eine Belegung $\gamma$ für $\varphi$ genau dann erfüllend ist, wenn die Variablenbelegung $X \mapsto \{2^x \mid x \in \gamma(X)\}$ erfüllend für $\varphi'$ ist.

Die Berechnung von $\varphi'$ ist in logarithmischem Raum möglich, da die eingegebenen Zahlen bereits in Binärdarstellung vorliegen.

Da wir bei unserer Konstruktion neben der Multiplikation auch die Vereinigung verwenden, ist

$$\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +\} \cup \mathrm{O}) \leq_{\mathrm{m}}^{\log} \mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times, \cup\} \cup \mathrm{O})$$

gezeigt. Nach Lemma 2.6 gilt

$$\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times, \cup\} \cup \mathrm{O}) \leq_{\mathrm{m}}^{\log} \mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times, -\}).$$

Dies vervollständigt den Beweis. □

Betrachte das folgende Problem.

**Definition 4.25**
Für $n \in \mathbb{N}$ ist $n$ ein Integer-Ausdruck. Für Integer-Ausdrücke $\alpha, \beta$ sind $(\alpha + \beta)$ und $(\alpha \cup \beta)$ Integer-Ausdrücke.
Wir definieren die von einem Integer-Ausdruck $\alpha$ beschriebene Menge $L(\alpha) \subseteq \mathbb{N}$:
Für $n \in \mathbb{N}$ ist $L(n) = \{n\}$. Für $(\beta \oplus \gamma)$ und $\oplus \in \{+, \cup\}$ gilt $L(\beta \oplus \gamma) = L(\beta) \oplus L(\gamma)$.

$$\mathrm{INEQ} =_{\mathrm{def}} \{(\alpha, \beta) \mid \alpha, \beta \text{ sind Integer-Ausdrücke und } L(\alpha) \neq L(\beta)\}.$$

Meyer und Stockmeyer [SM73] beweisen das folgende Lemma.

**Satz 4.26 ([SM73])**
INEQ *ist* $\leq_{\mathrm{m}}^{\log}$*-vollständig für* $\Sigma_2^{\mathrm{p}}$.

Wir nennen das Problem

$\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +, \cup\}) \cap \{\varphi \mid \varphi \text{ ist eine } \{+, \cup\}\text{-Formel, in der keine Variablen auftreten}\}$

$\mathrm{constCSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +, \cup\})$.

[21] Da die $Y_i$'s eigentlich nicht mehrfach auftreten müssten, enthält die entstehende Formel so einige redundante Abschnitte, kann aber noch immer in logarithmischem Raum berechnet werden, was für unsere Zwecke hinreichend ist.

**Satz 4.27**
$\mathrm{constCSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +, \cup\})$ *ist* $\leq_{\mathrm{m}}^{\log}$*-hart für* $\Pi_2^{\mathrm{p}}$.

*Beweis.* Nach Lemma 4.26 ist $\overline{\mathrm{INEQ}} \leq_{\mathrm{m}}^{\log}$-hart für $\Pi_2^{\mathrm{p}}$.
Seien $\alpha$, $\beta$ Integer-Ausdrücke. $\alpha$, $\beta$ sind somit $\{+, \cup\}$-Terme. Es gilt daher

$$(\alpha, \beta) \in \overline{\mathrm{INEQ}} \Leftrightarrow \alpha = \beta \in \mathrm{constCSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +, \cup\}).$$

□

Damit ist auch $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +, \cup\}) \leq_{\mathrm{m}}^{\log}$-hart für $\Pi_2^{\mathrm{p}}$. Es sollte aber möglich sein, eine bessere obere Schranke – wie etwa die $\leq_{\mathrm{m}}^{\log}$-Härte für $\Sigma_3^{\mathrm{p}}$ – zu zeigen, da wir bei der Reduktion keinen Gebrauch von Variablen gemacht haben, sondern ausschließlich Konstanten verwendet haben.
Wegen Lemma 4.24 ist ferner auch $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times, \cup\}) \leq_{\mathrm{m}}^{\log}$-hart für $\Pi_2^{\mathrm{p}}$. Damit haben wir das folgende Korollar bewiesen.

**Korollar 4.28**
*Es gilt:*

1. $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +, \cup\})$ *ist* $\leq_{\mathrm{m}}^{\log}$*-hart für* $\Pi_2^{\mathrm{p}}$.
2. $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times, \cup\})$ *ist* $\leq_{\mathrm{m}}^{\log}$*-hart für* $\Pi_2^{\mathrm{p}}$.

Sofern neben einer arithmetischen Operation noch eine der Operationen Schnitt und Mengendifferenz zur Verfügung steht, lässt sich die PSPACE-Härte zeigen. Wir modifizieren dazu einen Beweis von McKenzie und Wagner [MW03] geringfügig, sodass wir ohne die Vereinigung auskommen.

Wir benötigen für die weitere Argumentation das folgende als $\leq_{\mathrm{m}}^{\log}$-vollständig bekannte Problem 3KNF-QBF.

**Definition 4.29**
Es sei

$$\begin{aligned} \text{3KNF-QBF} =_{\mathrm{def}} \{F \mid &\varphi \text{ ist eine in 3-KNF vorliegende, abgeschlossene,} \\ &\text{quantifizierte boolesche Formel mit } F \equiv 1\}. \end{aligned}$$

**Satz 4.30**
*Es gelten die folgenden Aussagen:*

1. *Es ist* $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +, \cap\}\Big) \leq_{\mathrm{m}}^{\mathrm{p}}$*-hart für* PSPACE.
2. *Es ist* $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times, \cap\}\Big) \leq_{\mathrm{m}}^{\log}$*-hart für* PSPACE.
3. *Es ist* $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +, -\}\Big) \leq_{\mathrm{m}}^{\log}$*-hart für* PSPACE.
4. *Es ist* $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \times, -\}\Big) \leq_{\mathrm{m}}^{\log}$*-hart für* PSPACE.

*Beweis.* Wir zeigen die ersten beiden Aussagen jeweils mittels einer $\leq_{\mathrm{m}}^{\mathrm{p}}$- bzw. mittels einer $\leq_{\mathrm{m}}^{\log}$-Reduktion von 3KNF-QBF.

1. Es sei $F = Q_1 x_1 \dots Q_m x_m H(x_1, \dots, x_m)$ eine 3KNF-QBF-Instanz, wobei $H = \bigwedge_{j=1}^{n} H_j$ für ein $n \in \mathbb{N}$ sei, die Formeln $H_1, \dots, H_n$ Klauseln mit jeweils drei Literalen seien und $Q_1, \dots, Q_m \in \{\exists, \forall\}$ gelte.

Wir konstruieren im Folgenden für $k = m, m-1, \dots, 1, 0$ eine Folge von Konjunktionen von Atomen $\varphi_k$ mit Variablen $X_m, X_{m-1}, \dots, X_k$, sodass für alle $\alpha_1, \dots, \alpha_k \in \{0,1\}$ und jede Termbelegung $I$ die folgende Bedingung gilt:

$$\begin{aligned} &Q_{k+1} x_{k+1} \dots Q_m x_m H(\alpha_1, \dots, \alpha_k, x_{k+1}, \dots, x_m) \Leftrightarrow \\ &\Leftrightarrow \sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{k} \alpha_i 6^{n+i} + \sum_{i=k+1}^{m} 6^{n+i} \in I(X_k) \qquad (*) \end{aligned}$$

Die $\varphi_k$ werden dabei so aufgebaut, dass für je zwei Termbelegungen $I, I'$

$$I(X_k) = I'(X_k)$$

gilt. Aus diesem Grunde werden wir auch einfach $X_k$ für die Menge $I(X_k)$ für eine beliebige Termbelegung $I$ schreiben.
Aus $(*)$ erhalten wir für $k = 0$

$$Q_1 x_1 \dots Q_m x_m H(x_1, \dots, x_m) \Leftrightarrow \sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{m} 6^{n+i} \in X_0.$$

Damit ergibt sich für $r = \sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{m} 6^{n+i}$ mit

$$F \mapsto \exists X_m \exists X_{m-1} \dots \exists X_0 : \varphi_0 \wedge \big(X_0 \cap \{r\} = \{r\}\big)$$

eine – wie aus dem Folgenden hervorgehen wird – in polynomieller Zeit berechenbare Reduktionsfunktion für 3KNF-QBF $\leq_{\mathrm{m}}^{\mathrm{p}}$ $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +, \cap\}\Big)$.

Wir betrachten zunächst den Fall $k = m$. Definiere

$$a_i =_{\mathrm{def}} 6^{n+i} + \sum_{x_i \text{ in } H_j} 6^j \text{ und } b_i =_{\mathrm{def}} \sum_{\overline{x_i} \text{ in } H_j} 6^j$$

für $i = 1, \dots, m$ und setze

$$\varphi_m =_{\mathrm{def}} \Big(X_m = \sum_{i=1}^{m} \{a_i, b_i\} + \sum_{j=1}^{n} \{0, 6^j, 2 \cdot 6^j\}\Big).$$

Man beachte dabei, dass jeder Summand $6^j$ in dem obigen Ausdruck höchstens 5 mal auftritt.
Wir zeigen, dass dann für alle $\alpha_1, \dots, \alpha_m \in \{0,1\}$

$$H(\alpha_1, \dots, \alpha_m) \Leftrightarrow \sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{m} \alpha_i \cdot 6^{n+i} \in X_m$$

gilt:

„⇒“: Es sei $\gamma_i \in \{a_i, b_i\}$ mit $\gamma_i = a_i$ für $\alpha_i = 1$ und $\gamma_i = b_i$ sonst. Da $H(\alpha_1, \ldots, \alpha_m)$ wahr ist, gibt es für jede Klausel $H_j$ mindestens ein wahres Literal und somit gibt es ein $\gamma_i$, in welchem der Summand $6^j$ auftritt. Daher ist

$$\sum_{i=1}^{m} \gamma_i = \sum_{j=1}^{n} \beta_j 6^j + \sum_{i=1}^{m} \alpha_i 6^{n+i}$$

mit $\beta_j \in \{1, 2, 3\}$. Dann gilt

$$\sum_{i=1}^{m} \gamma_i + \sum_{j=1}^{n} (3 - \beta_j) \cdot 6^j = \sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{m} \alpha_i \cdot 6^{n+i} \in X_m.$$

“⇐“: Es sei

$$\sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{m} \alpha_i \cdot 6^{n+i} \in X_m.$$

Dann ist

$$\sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{m} \alpha_i \cdot 6^{n+i} \in \sum_{i=1}^{m} \gamma_i + \sum_{j=1}^{n} \{0, 6^j, 2 \cdot 6^j\}$$

für

$$\gamma_i = \begin{cases} a_i & \alpha_i = 1 \\ b_i & \alpha_i = 0 \end{cases}.$$

Da für jede Klausel $H_j$ der Summand $6^j$ in $\sum_{i=1}^{m} \gamma_i$ mindestens einmal auftritt, gilt wegen der Wahl der $\gamma_i$

$$H(\alpha_1, \ldots, \alpha_m).$$

Für den Schritt von $k$ auf $k-1$ setzen wir $(*)$ voraus. Für den Fall $Q_k = \exists$ erhalten wir

$$\begin{aligned}
&\exists x_k Q_{k+1} \ldots Q_m x_m H(\alpha_1, \ldots, \alpha_{k-1}, x_k, x_{k+1}, \ldots, x_m) \\
&\Leftrightarrow Q_{k+1} x_{k+1} \ldots Q_m x_m H(\alpha_1, \ldots, \alpha_{k-1}, 0, x_{k+1}, \ldots, x_m) \vee \\
&\qquad \vee Q_{k+1} x_{k+1} \ldots Q_m x_m H(\alpha_1, \ldots, \alpha_{k-1}, 1, x_{k+1}, \ldots, x_m) \\
&\overset{\text{IV}}{\Leftrightarrow} \sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{k-1} \alpha_i 6^{n+i} + \sum_{i=k+1}^{m} 6^{n+i} \in X_k \vee \sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{k-1} \alpha_i 6^{n+i} + \sum_{i=k}^{m} 6^{n+i} \in X_k \\
&\Leftrightarrow \sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{k-1} \alpha_i 6^{n+i} + \sum_{i=k}^{m} 6^{n+i} \in X_{k-1},
\end{aligned}$$

wobei $\varphi_{k-1} =_{\text{def}} \varphi_k \wedge \Big(X_{k-1} = \big(X_k + \{0, 6^{n+k}\}\big)\Big)$.

Im Fall $Q_k = \forall$ lässt sich analog argumentieren:

$$\begin{aligned}
&\forall x_k Q_{k+1} \ldots Q_m x_m H(\alpha_1, \ldots, \alpha_{k-1}, x_k, x_{k+1}, \ldots, x_m) \\
&\Leftrightarrow Q_{k+1} x_{k+1} \ldots Q_m x_m H(\alpha_1, \ldots, \alpha_{k-1}, 0, x_{k+1}, \ldots, x_m) \wedge \\
&\qquad \wedge Q_{k+1} x_{k+1} \ldots Q_m x_m H(\alpha_1, \ldots, \alpha_{k-1}, 1, x_{k+1}, \ldots, x_m) \\
&\overset{\text{IV}}{\Leftrightarrow} \sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{k-1} \alpha_i 6^{n+i} + \sum_{i=k+1}^{m} 6^{n+i} \in X_k \wedge \sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{k-1} \alpha_i 6^{n+i} + \sum_{i=k}^{m} 6^{n+i} \in X_k \\
&\Leftrightarrow \sum_{j=1}^{n} 3 \cdot 6^j + \sum_{i=1}^{k-1} \alpha_i 6^{n+i} + \sum_{i=k}^{m} 6^{n+i} \in X_{k-1},
\end{aligned}$$

wobei $\varphi_{k-1} =_{\text{def}} \varphi_k \wedge \Big(X_{k-1} = (X_k + \{6^{n+k}\}) \cap X_k\Big)$.

Offenbar kann die Reduktionsfunktion in polynomieller Zeit berechnet werden.

2. Es lässt sich 3KNF-QBF $\leq_{\text{m}}^{\log}$ CSP$\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, \times, \cap\}\Big)$ analog zeigen: Es sind lediglich die Operationen + durch $\times$ und die Zahlen $6^i$ für $i = 1, \ldots, n+m$ durch $p_i$ zu ersetzen, wobei $p_i$ die $i$-te Primzahl ist. Wegen $\pi(i) \in \Theta(i/\log i)$ gilt $p_{n+m} \in O((n+m)^2)$ und daher $|p_{n+m}| \in O(\log(n+m))$. Die auftretenden Produkte sind nicht innerhalb der Reduktion auszurechnen, sondern als Produkte in die auszugebende Formel zu schreiben.
Ansonsten lässt sich überall wie in Teil 1 argumentieren.

3. Für CSP$\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, -\}\Big)$ lässt sich wie für CSP$\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, \cap\}\Big)$ vorgehen, der Schnitt kann gemäß dem Beweis von Lemma 2.6 durch die Mengendifferenz simuliert werden. Es kann dabei aber sogar die $\leq_{\text{m}}^{\log}$-Härte für PSPACE gezeigt werden:

Die in der Ausgabeformel auftretenden Konstanten müssen nicht innerhalb der Reduktionsfunktion berechnet werden, sondern können mittels der Shift-And-Add-Technik innerhalb der Ausgabeformel beschrieben werden. Dabei sind lediglich Zahlen der Form $6^k$ für $k \leq n+m$ zu beschreiben. Da $n$ und $m$ als Anzahl der Klauseln bzw. Variablen in der Eingabeformel codiert sind, kann die Binärdarstellung von $n+m$ und kleineren Zahlen in logarithmischem Raum berechnet werden.

Mehrelementige Mengen können erzeugt werden, da wir mit der Mengendifferenz die Vereinigung simulieren können (vgl. dazu den Beweis von Lemma 2.6).

4. folgt aus 2. mit Lemma 2.6. □

In Abschnitt 3 hatten wir festgestellt, dass es CSPs gibt, für welche die Variante über $[\mathbb{N}]$ schwerer als die Variante über $\mathcal{P}_{\text{fin}}(\mathbb{N})$ ist, aber auch bereits angekündigt, dass es Situationen gibt, in welchen die umgekehrte Situation vorliegt. An dieser Stelle können wir dieses Versprechen einlösen.

Wir hatten in Korollar 4.23 die $\leq_{\text{m}}^{\log}$-Vollständigkeit von CSP$\Big([\mathbb{N}], \{=, +, \cap\}\Big)$ für NP festgestellt. Nach Satz 4.30 ist hingegen CSP$\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, \cap\}\Big)$ $\leq_{\text{m}}^{\text{p}}$-hart für PSPACE. Unter der Annahme NP $\neq$ PSPACE gilt daher

$$\text{CSP}\Big([\mathbb{N}], \{=, +, \cap\}\Big) \leq_{\text{m}}^{\text{p}} \text{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, \cap\}\Big)$$

aber

$$\text{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, \cap\}\Big) \not\leq_{\text{m}}^{\text{p}} \text{CSP}\Big([\mathbb{N}], \{=, +, \cap\}\Big).$$

## 4.5 Diskussion Zweier Offen Bleibender Fragen

Bei den bislang untersuchten Problemen bleiben einige offene Fragestellungen, von denen wir an dieser Stelle auf die zwei aus unserer Sicht grundlegendsten eingehen. Wir geben in diesem Abschnitt Aufschluss darüber, wie wir versucht haben, diese – zumindest teilweise – zu klären, und versuchen, möglichst genau zu begründen, wo die wesentlichen Schwierigkeiten bei den beiden Fragestellungen liegen.

1. Bei den CSPs mit $O \in \{\{+\}, \{\times\}\}$ hatten wir für die Variante über endlichen Intervallen die NP-Vollständigkeit bzw. die NP-Härte und Zugehörigkeit zu $\Sigma_3^{\mathrm{p}}$ festgestellt. Für die Variante über beliebigen endlichen Mengen konnten wir hingegen lediglich die $\leq_{\mathrm{m}}^{\log}$-Härte für NP und die Zugehörigkeit zu NEXP zeigen. Die beiden Schranken liegen also sehr weit auseinander.

   Wir werden hier jedoch lediglich den Fall $O = \{+\}$ weiter erörtern. Teilweise lassen sich die Erörterungen für den zweiten Fall analog tätigen.

2. Für $\mathrm{CSP}\Big([\mathbb{N}], \{=, +, \cap\}\Big)$ konnten wir wiederum die NP-Vollständigkeit zeigen, für $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +, \cap\}\Big)$ hingegen nicht einmal die Entscheidbarkeit. Die Zugehörigkeit zu $\Sigma_1$ ergibt sich aus Satz 4.1.

**Arbeiten von Jeż und Okhotin:** Jeż und Okhotin [JO10a] untersuchten ähnliche Fragestellungen wie wir. Sie betrachteten unter anderem Gleichungen über Variablen, Konstanten und der Addition, wobei die Variablen beliebige Mengen natürlicher Zahlen als Werte annehmen dürfen, während Konstanten „ultimately periodic“ Mengen sind, d.h. Mengen $A \subseteq \mathbb{N}$, für die es $d, p \in \mathbb{N}$ gibt, sodass für alle $x \geq d$

$$x \in A \Leftrightarrow x + p \in A$$

gilt. Unter anderem untersuchten sie die Komplexität des Problems, zu testen, ob es für ein derartiges Gleichungssystem eine Lösung gibt und stellten dabei die $\Pi_1$-Vollständigkeit fest.
Für unsere Zwecke hilft dieses Resultat nicht direkt weiter, da Jeż und Okhotin Probleme untersuchen, die einerseits aufgrund der zugelassenen unendlichen Mengen als Konstanten schwerer zu lösen sind als unsere CSPs, andererseits jedoch in gewissem Sinne einfacher zu lösen sind, da Variablen nicht nur mit endlichen Teilmengen von $\mathbb{N}$ belegt werden können. Ferner konnten wir bereits $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +\}\Big) \in \mathrm{NEXP}$ zeigen, was eine wesentlich bessere obere Schranke darstellt als $\Pi_1$.

In weiteren Arbeiten [JO10b, JO11, JO14] erlauben Jeż und Okhotin auch weitere Operationen neben der Addition, nämlich Schnitt und Vereinigung, aber die Resultate lassen sich aus den gleichen Gründen nicht auf unsere Situation übertragen.

Generell gilt, dass Jeż und Okhotin mit ihren Untersuchungen keinen unserer Bereiche abdecken [Jez15]. Insbesondere sind ihre bisherigen Arbeiten auf Gleichungssysteme beschränkt, die kleinste/größte/eindeutige Lösungen haben [Jez15].

**Sumsets:** Im Folgenden werden wir die Möglichkeit $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, +\}\Big) \notin \mathrm{EXP}$ diskutieren.

**Definition 4.31**
Sei $(G, +)$ eine abelsche Gruppe und $S \subseteq G$ endlich. Dann heißt $S$ genau dann **sumset**, wenn es ein $A \subseteq G$ mit $A + A = S$ gibt.

Croot und Lev [CL07] weisen darauf hin, dass nicht bekannt ist, ob es einen Polynomialzeitalgorithmus gibt, der für eine gegebene Menge entscheidet, ob sie ein sumset ist. Genauer erwähnen sie, dass der beste bekannte Algorithmus für jedes $s \in S$ und für alle $2^{|S|}$ Teilmengen $S''$ der Menge $S' = \{s' - s \mid s' \in S\}$ jeweils $S'' + S''$ berechnet und mit $S'$ vergleicht.
Für den Fall $G = \mathbb{Z}$ ist es notwendig und hinreichend, lediglich die Elemente $s \in S \cap 2\mathbb{Z}$ zu betrachten. Für einen sumset $S$ sind nämlich $\min(S)$ und $\max(S)$ stets gerade. Weiterhin ist $[1, 3]$ kein sumset, $\{s - 1 \mid s \in [1, 3]\}$ hingegen schon.

Zwar formulieren Croot und Lev das Problem allgemein über Teilmengen abelscher Gruppen, erwähnen jedoch explizit, dass auch dann, wenn die Gruppe zyklisch ist, kein besserer Algorithmus bekannt ist. Insbesondere ist also kein besserer Algorithmus zum Testen, ob eine endliche Menge $S \subseteq \mathbb{Z}$ ein sumset ist, bekannt.

**Lemma 4.32**
*Sei $A \subseteq \mathbb{Z}$ und $z \in \mathbb{Z}$. Dann ist $A$ genau dann ein sumset, wenn $A + \{2z\}$ ein sumset ist.*

*Beweis.* Es gilt für $X \subseteq \mathbb{Z}$

$$X + X = A \Leftrightarrow (X + \{z\}) + (X + \{z\}) = A + \{2z\}.$$

□

Statt zu testen, ob $S \subseteq \mathbb{Z}$ ein sumset ist, kann also auch getestet werden, ob

$$S' = \left\{ s - 2\left\lfloor \frac{\min(S \cup \{0\})}{2} \right\rfloor \mid s \in S \right\} \subseteq \mathbb{N}$$

ein sumset ist.
Daher ist nicht bekannt, ob SUMSET $\in$ P gilt, wobei

$$\text{SUMSET} =_{\text{def}} \{S \in \mathcal{P}_{\text{fin}}(\mathbb{N}) \mid S \text{ ist ein sumset}\}.$$

SUMSET $\in$ NP ist hingegen offensichtlich. Weiter gilt offenbar

$$\text{SUMSET} \leq_{\text{m}}^{\log} \text{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +\}\Big).$$

Mit $\{+\}$-Termen der Länge $n$ können jedoch sogar Mengen $S$ beschrieben werden, sodass

$$|S| \in 2^{\Theta(n)}$$

gilt.

Wir definieren

$$\text{SUCCINCT-SUMSET} =_{\text{def}} \{(C_1, \ldots, C_n) \mid C_i \in \mathcal{P}_{\text{fin}}(\mathbb{N}), \sum_{i=1}^{n} C_i \in \text{SUMSET}\}.$$

Offenbar gilt dann SUCCINCT-SUMSET $\leq_{\text{m}}^{\log}$ CSP$\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +\}\Big)$.

Für SAT lässt sich ebenfalls eine Variante SUCCINCT-SAT definieren. Dabei ist SAT vollständig für NP und SUCCINCT-SAT vollständig für NEXP. Mittels Translationslemma gilt

$$\text{SAT} \in \text{P} \Rightarrow \text{SUCCINCT-SAT} \in \text{EXP}.$$

Falls also EXP $\neq$ NEXP gilt, so gilt nach dem Translationslemma auch P $\neq$ NP. Aus EXP $\neq$ NEXP folgt daher SAT $\notin$ P und SUCCINCT-SAT $\notin$ EXP.
Nach derzeitigem Kenntnisstand wäre es möglich, dass für die beiden Probleme SUMSET und SUCCINCT-SUMSET die gleiche Situation vorliegt, dass also – falls EXP $\neq$ NEXP – sowohl SUMSET $\notin$ P als auch SUCCINCT-SUMSET $\notin$ EXP gilt.

**Obere Schranke für CSP$\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, \cap\}\Big)$:** Wir wissen bereits aus Satz 4.30, dass CSP$\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, \cap\}\Big)$ $\leq_{\text{m}}^{\log}$-hart für PSPACE ist. Als obere Schranke konnten wir bislang lediglich die triviale Eigenschaft CSP$\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, \cap\}\Big) \in \Sigma_1$ zeigen.
Wir versuchen Hinweise darauf zu finden, dass das Beweisen einer besseren oberen Schranke schwer ist:
Wir konnten CSP$\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +\}\Big) \in$ NEXP durch einen Algorithmus zeigen, der zunächst notwendige Variablenbelegungen bestimmt bzw. aus einer endlichen Menge rät und anschließend die verbliebenen Variablen mit der leeren Menge belegt. Mit Hilfe der Schnittoperation können wir jedoch ausdrücken, dass eine Variable $X$ unter einer erfüllenden Variablenbelegung $\alpha$ nicht mit $\emptyset$ belegt werden kann, ohne jedoch damit die Größe der Menge $\alpha(X)$ für eine gültige Variablenbelegung $\alpha$ zu beschränken. Ein Beispiel dafür stellt der folgende Satz dar:

$$\exists X \exists Y \ (X \cap Y = \{0\}) \wedge (X + \{1\} = X).$$

Offenbar hat dieser Satz keine erfüllende Belegung über $\mathcal{P}_{\text{fin}}(\mathbb{N})$. Das Atom $X + \{1\} = X$ würde nur von einer Belegung $\alpha$ mit $\alpha(X) = \emptyset$ erfüllt werden. Eine derartige Belegung erfüllt jedoch das erste Atom nicht.
Ein iterativer Algorithmus der Art wie in Lemma 4.2, der versucht, sukzessive eine erfüllende Belegung $\alpha$ zu konstruieren, würde zunächst $\alpha(X) \supseteq \{0\}$, dann $\alpha(X) \supseteq \{0, 1\}$, usw. festlegen. In diesem Fall würde er nicht terminieren. Falls jedoch eine Abbruchbedingung der Art wie etwa in

$$\exists X \exists Y \ (X \cap Y = \{0\}) \wedge \big((X + \{1\}) \cap \sum_{i=1}^{k} [0, c_i] = X\big)$$

vorliegt, findet er eine erfüllende Belegung.
Es bleibt offen, ob für einen solchen iterativen Algorithmus eine Schranke existiert, sodass er stets entweder terminiert, ohne Belegungen oberhalb dieser Schranke vorzunehmen, oder gar nicht terminiert. Falls eine solche Schranke existiert, wäre CSP$\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, +, \cap\}\Big)$ entscheidbar.

Wir betrachten ein weiteres Beispiel. Seien $C_1, \ldots, C_k$ und $C'_1, \ldots, C'_{k'}$ beliebige Konstanten und $\varphi(C_1, \ldots, C_k)$ sowie $\psi(C'_1, \ldots, C'_{k'})$ Terme über $\{+, \cap\}$. Betrachte den Satz

$$\exists X \exists Y \ \big(X = Y + Y\big) \wedge \big(X \cap \varphi(C'_1, \ldots, C'_{k'}) = \psi(C_1, \ldots, C_k)\big).$$

Er formuliert die Frage: Gibt es einen sumset $X$ mit $\psi(C_1, \ldots, C_k) \subseteq X$ und

$$\big(\varphi(C'_1, \ldots, C'_{k'}) - \psi(C_1, \ldots, C_k)\big) \cap X = \emptyset?$$

Es muss also die Frage nach der Existenz eines sumsets $S$, für den exponentiell viele Forderungen der Form $x \in S$ oder $x \notin S$ für polynomiell lange $x$ gegeben sind, beantwortet werden.
Hier ist zunächst unklar, ob im Falle der Existenz eines solchen sumsets auch immer ein „kleiner“ solcher sumset existiert, oder ob der kleinste derartige sumset „sehr groß“ sein kann.

## 4.6 Addition und Multiplikation

Sobald in einem CSP sowohl die Addition als auch die Multiplikation zugelassen ist, erhalten wir unentscheidbare, aber aufzählbare Probleme. Es ist leicht zu sehen, dass wir beliebige diophantische Gleichungen formulieren können.

Genauer gilt: Wir zeigen die $\leq_{\mathrm{m}}^{\log}$-Vollständigkeit solcher CSPs für $\Sigma_1$[22].

**Satz 4.33**
*Es sei* $M \in \{\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), [\mathbb{N}]\}$. *Dann ist* $\mathrm{CSP}\big(\mathrm{M}, \{=, +, \times\} \cup \mathrm{O}\big) \leq_{\mathrm{m}}^{\log}$*-vollständig für* $\Sigma_1$, *wobei* $O \subseteq \{\cup, \cap, -\}$.

*Beweis.* Wir zeigen zunächst:
A) Für einen beliebigen $\{+, \times\}$-Satz $\psi$ kann in logarithmischem Raum ein $\{+, \times\}$-Satz $\psi'$ berechnet werden, sodass

$$\psi' \in \mathrm{CSP}\big(\mathrm{M}, \{=, +, \times\}\big) \Leftrightarrow \text{es existiert für } \psi \text{ eine erfüllende Variablenbelegung mit Wertebereich } \{\{a\} \mid a \in \mathbb{N}\}$$

gilt. Wir geben an, wie $\psi'$ aus $\psi$ konstruiert wird:

Hänge für jede Variable $X$ in $\psi$ die folgende Konjunktion von Atomen an das Ende von $\psi$ an:

$$\wedge\Big(X + X = \{2\} \times X\Big) \wedge \Big(X \times \{0\} = \{0\}\Big).$$

Sei $\alpha$ eine beliebige Termbelegung für $\psi'$. Wäre $\alpha(X) = \emptyset$, so wäre das Atom $X \times \{0\} = \{0\}$ nicht erfüllt. Wäre $\alpha(X) = \{x_1, \ldots, x_r\}$ für ein $r \in \mathbb{N} - \{0, 1\}$ und $x_i < x_{i+1}$, so enthielte $\alpha(X + X) = \alpha(X) + \alpha(X) \supseteq \{x_1 + x_1, x_1 + x_2, \ldots, x_1 + x_r, x_r + x_r\}$ mindestens $r + 1$ Elemente, während $\alpha(\{2\} \times X) = \{2\} \times \alpha(X)$ lediglich $r$ Elemente enthielte. Es wäre folglich $\alpha$ nicht erfüllend.
Jede erfüllende Termbelegung für $\psi'$ ordnet somit jeder Variable eine einelementige Menge zu. Ferner ist jede erfüllende Variablenbelegung für $\psi'$ auch erfüllend für $\psi$.

Existiert umgekehrt eine erfüllende Variablenbelegung für $\psi$, die jeder Variablen eine einelementige Menge zuweist, so ist diese offenbar auch für $\psi'$ erfüllend.

B) Dem Matiyasevich-Robinson-Davis-Putnam-Theorem [Mat70, DPR61] zufolge existiert ein $n \in \mathbb{N}$ und ein multivariates Polynom $p$ mit ganzzahligen Koeffizienten, sodass für jede Menge $A \in \Sigma_1$ ein $a \in \mathbb{N}$ existiert, sodass

$$x \in A \Leftrightarrow \exists y \in \mathbb{N}^n, p(a, x, y) = 0$$

gilt.

Wir können nun in der diophantischen Gleichung $p(a, x, y) = 0$ die Monome mit negativen Koeffizienten auf die rechte Seite ziehen. Damit existieren multivariate Polynome $l, r$ mit natürlichzahligen Koeffizienten, sodass

$$x \in A \Leftrightarrow \exists y \in \mathbb{N}^n \; l(a, x, y) = r(a, x, y)$$

gilt.

[22] Dass wir die $\leq_{\mathrm{m}}^{\log}$-Vollständigkeit zeigen, mag auf den ersten Blick etwas seltsam anmuten. Wir tun dies jedoch aus dem Grunde, dass dadurch die Übersichtstabelle im Abschnitt 4.7 „schöner“ wird.

Wegen A) können wir die rechte Seite der obigen Äquivalenz als einen $\{+, \times\}$-Satz $\varphi$ formulieren.

Dabei sind $p$ und $a$ unabhängig von der Eingabe $x$. Da insbesondere für im Polynom auftretende Terme der Form $x^e$ der Exponent $e$ konstante Größe hat, kann der Satz $\varphi$ in logarithmischem Raum berechnet werden.

Es gilt $\mathrm{CSP}\big(\mathrm{M}, \{=, +, \times\} \cup \mathrm{O}\big) \in \Sigma_1$ wegen Satz 4.1. □

## 4.7 Übersicht und Fazit

Folgende Tabellen liefern eine Übersicht über die in diesem Kapitel gewonnenen Resultate. Die erste Tabelle behandelt die CSPs über $\mathcal{P}_{\text{fin}}(\mathbb{N})$. Hier tritt auch der einzige Fall auf, in welchem wir eine untere Schranke nicht hinsichtlich der $\leq_{\text{m}}^{\log}$- sondern lediglich hinsichtlich der $\leq_{\text{m}}^{\text{p}}$-Reduktion zeigen konnten.

| $\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \text{O}\big)$ mit $O =$ | Härte | enthalten in |
|---|---|---|
| $\{+\}$ | $\leq_{\text{m}}^{\log}$-hart für NP, 4.7 | NEXP, 4.3 |
| $\{\times\}$ | $\leq_{\text{m}}^{\log}$-hart für NP, 4.20 | NEXP, 4.19 |
| $\{+, \cap\}$ | $\leq_{\text{m}}^{\text{p}}$-hart für PSPACE, 4.30 | $\Sigma_1$, 4.1 |
| $\{+, \cup\}$ | $\leq_{\text{m}}^{\log}$-hart für $\Pi_2^{\text{p}}$, 4.28 | $\Sigma_1$, 4.1 |
| $\{+, -\}$ | $\leq_{\text{m}}^{\log}$-hart für PSPACE, 4.30 | $\Sigma_1$, 4.1 |
| $\{+, \times\}$ | $\leq_{\text{m}}^{\log}$-hart für $\Sigma_1$, 4.33 | $\Sigma_1$, 4.1 |

Die nachfolgende Tabelle gibt Aufschluss über die CSPs über $[\mathbb{N}]$.

| $\text{CSP}\big([\mathbb{N}], \{=\} \cup \text{O}\big)$ mit $O =$ | $\leq_{\text{m}}^{\log}$-hart für | enthalten in |
|---|---|---|
| $\{+\}$ | NP, 4.7 | NP, 4.3 |
| $\{\times\}$ | NP, 4.20 | $\Sigma_3^{\text{p}}$, 4.13 |
| $\{+, \cap\}$ | NP, 4.23 | NP, 4.22 |
| $\{+, \times\}$ | $\Sigma_1$, 4.33 | $\Sigma_1$, 4.1 |

Die Frage nach der Komplexität von CSPs, in denen auch arithmetische Operationen erlaubt sind, konnte in dieser Arbeit nicht erschöpfend geklärt werden. Vielmehr stellt sich heraus, dass schon beim Zulassen weniger Operationen die Probleme sehr schwer werden. Insbesondere liegt Unentscheidbarkeit vor, sobald die beiden arithmetischen Operationen zur Verfügung stehen.

Die Frage nach der Entscheidbarkeit von Problemen konnte nur für einzelne Probleme beantwortet werden.

Trotz der Unvollständigkeit der gefundenen Resultate lässt sich für die oben aufgeführten Probleme eine Tendenz feststellen: Sofern $[\mathbb{N}]$ abgeschlossen bezüglich der zugelassenen Operationen ist, werden die Probleme durch das Einschränken der Konstanten und Variablen auf endliche Intervalle über $\mathbb{N}$ einfacher.
Lässt man die Vereinigung oder die Mengendifferenz zu, unterscheiden sich die Probleme über $\mathcal{P}_{\text{fin}}(\mathbb{N})$ und $[\mathbb{N}]$ hinsichtlich der Konstanten nicht mehr. Der einzige Unterschied besteht dann darin, ob die Variablen mit beliebigen endlichen Mengen oder nur mit endlichen Intervallen belegt werden dürfen.
Wenn die Konstanten beliebige endliche Teilmengen von $\mathbb{N}$ sind, dann ist es natürlicher, auch die Variablen mit Werten aus $\mathcal{P}_{\text{fin}}(\mathbb{N})$ zu belegen.

Zusammenfassend lässt sich sagen: Es bleiben vor allem bei den CSPs über endlichen Teilmengen der natürlichen Zahlen einige Probleme offen, die weitere intensive Beschäftigung verlangen.

# Literatur


[BGSW05] E. Böhler, C. Glaßer, B. Schwarz, and K. W. Wagner. Generation problems. *Theor. Comput. Sci.*, 345(2-3):260–295, 2005.

[CL07] E. S. Croot, III and V. F. Lev. Open problems in additive combinatorics. In *Additive combinatorics*, volume 43 of *CRM Proc. Lecture Notes*, pages 207–233. Amer. Math. Soc., Providence, RI, 2007.

[DPR61] M. Davis, H. Putnam, and J. Robinson. The decision problem for exponential Diophantine equations. *Annals of Mathematics*, 74(2):425–436, 1961.

[FV99] T. Feder and M. Y. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, February 1999.

[GHR91] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. A Compendium of Problems Complete for P, 1991.

[GHR+07] C. Glaßer, K. Herr, C. Reitwießner, S. D. Travers, and M. Waldherr. Equivalence problems for circuits over sets of natural numbers. In Volker Diekert, Mikhail V. Volkov, and Andrei Voronkov, editors, *CSR*, volume 4649 of *Lecture Notes in Computer Science*, pages 127–138. Springer, 2007.

[GJM15] C. Glaßer, P. Jonsson, and B. Martin. Constraint satisfaction problems around skolem arithmetic. *CoRR*, abs/1504.04181, 2015.

[GRTW10] C. Glaßer, C. Reitwießner, S. Travers, and M. Waldherr. Satisfiability of algebraic circuits over sets of natural numbers. *Discrete Applied Mathematics*, 158(13):1394 – 1403, 2010.

[Jez15] A. Jez. Private Kommunikation, 2015.

[JO10a] A. Jez and A. Okhotin. On equations over sets of integers. *CoRR*, abs/1001.2932, 2010.

[JO10b] A. Jez and A. Okhotin. Univariate equations over sets of natural numbers. *Fundam. Inform.*, 104(4):329–348, 2010.

[JO11] A. Jez and A. Okhotin. Complexity of equations over sets of natural numbers. *Theory Comput. Syst.*, 48(2):319–342, 2011.

[JO14] A. Jez and A. Okhotin. Computational completeness of equations over sets of natural numbers. *Inf. Comput.*, 237:56–94, 2014.

[Mat70] Y. V. Matiyasevich. Enumerable sets are Diophantine. *Doklady Akad. Nauk SSSR*, 191:279–282, 1970. Translation in Soviet Math. Doklady, 11:354–357, 1970.

[MW03] P. McKenzie and K. W. Wagner. The complexity of membership problems for circuits over sets of natural numbers. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '03, pages 571–582, London, UK, UK, 2003. Springer-Verlag.

[Pap94] C. M. Papadimitriou. *Computational complexity.* Addison-Wesley, Reading, Massachusetts, 1994.

[Rei05] O. Reingold. Undirected st-connectivity in log-space. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 376–385, New York, NY, USA, 2005. ACM.

[SM73] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time(preliminary report). In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC '73, pages 1–9, New York, NY, USA, 1973. ACM.

**Versicherung des selbstständigen Arbeitens**

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe. Alle verwendeten Hilfsmittel und Quellen habe ich angegeben. Die Arbeit wurde von mir weder bisher noch gleichzeitig einer anderen Prüfungsbehörde mit der Folge des Verleihens eines akademischen Grades vorgelegt.

Würzburg, den 21.12.2015, _____

(Titus Dose)