Electronic Colloquium on Computational Complexity, Report No. 31 (2016)

Complexity of Constraint Satisfaction Problems over Finite Subsets of Natural Numbers

Titus Dose

Julius-Maximilians-Universität, Würzburg, Germany

Abstract

We study the computational complexity of constraint satisfaction problems that are based on integer expressions and algebraic circuits. On input of a finite set of variables and a finite set of constraints the question is whether the variables can be mapped onto finite subsets of $\mathbb{N}$ (resp., finite intervals over $\mathbb{N}$) such that all constraints are satisfied. According to the operations allowed in the constraints, the complexity varies over a wide range of complexity classes such as L, P, NP, PSPACE, NEXP, and even Σ_1, the class of c.e. languages.

ISSN 1433-8092

Contents

1 Introduction

The problems investigated in this paper are motivated by constraint satisfaction problems and integer expressions. We first introduce these notions and then explain their connection.

Constraint satisfaction problems. A constraint satisfaction problem (CSP) is a computational problem that on input of a finite set of variables and a finite set of constraints asks whether there is a mapping from the variables to some fixed domain such that all constraints are satisfied.
An example of a a classical CSP is 3-Colorability, i.e., the question of whether there is a mapping α from a graph's vertices onto $\{0, 1, 2\}$ such that for adjacent nodes u and v it holds that $\alpha(u) \neq \alpha(v)$.

The set of relations permitted in the constraints is called *constraint language*. Obviously CSPs over finite domains permitting arbitrary constraints belong to NP. The question which constraint languages lead to CSPs even decidable in polynomial time has been a topic of intensive research over the past decades.
Feder and Vardi [FV99] conjectured a dichotomy for CSPs over finite domains such that these CSPs are either in P or NP-complete. This conjecture is still open.

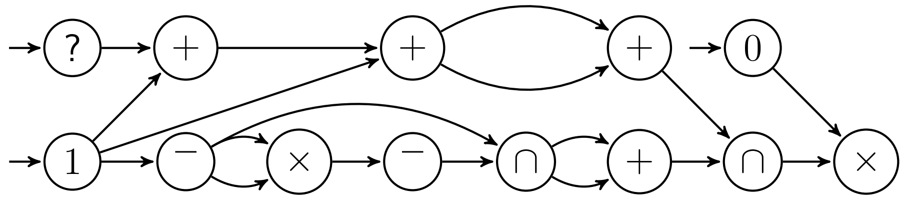
In the past years there has been an increasing interest in CSPs over *infinite* domains. Here much higher complexities are obtained, and some problems are even undecidable.

Integer expressions and algebraic circuits. In 1973, Meyer and Stockmeyer [SM73] asked for the complexity of decision problems regarding so-called integer expressions. An integer expression is a term built by singleton sets of natural numbers, the pairwise addition, and set operations like union, intersection, or complement. Meyer and Stockmeyer investigated the membership problem, i.e., the question of whether a given natural number is contained in the subset of $\mathbb{N}$ described by an integer expression. Moreover, they studied the inequality problem, i.e., the question of whether two given integer expressions describe the same subset of $\mathbb{N}$.
For some constant $m \in \mathbb{N}$ the integer expression $(0 \cup 2^1) + (0 \cup 2^2) + \cdots + (0 \cup 2^{m-1})$ describes the set of all even natural numbers $< 2^m$ in a succinct way. Note that here the natural number n is an abbreviation for the set $\{n\}$.

McKenzie and Wagner [MW03] considered generalized integer expressions and the complexity of membership problems for circuits over finite subsets of $\mathbb{N}$.
Here a circuit is a directed, acyclic graph with two kinds of nodes: on the one hand, there are input nodes containing a single natural number. On the other hand all remaining nodes, so-called operation nodes, perform one of the following operations: union, intersection, complement, pairwise addition, and pairwise multiplication. Each operation node has an indegree equal to the number of operands required by the operation.
So each of the nodes computes a set of natural numbers: the input nodes compute singletons, and each operation node computes the corresponding set obtained from its predecessors. The set computed by the circuit overall is defined as the set computed by some fixed output node.

In contrast to integer expressions, these circuits are able to store intermediate results and reuse them several times. Thus, it is possible to describe large numbers and sets in a succinct way.

Similar to Meyer and Stockmeyer, who investigated the equivalence problem for integer expressions, Glaßer et al. [GHR+07] considered the equivalence problem for circuits.
Moreover, Glaßer et al. [GRTW10] studied the satisfiability problem for circuits where the corresponding circuits are allowed to have unassigned input nodes. Now the problem is to decide for a given natural number b whether there exists an assignment for the input nodes such that b is contained in the set computed by the circuit.
This modification makes the circuits even more similar to CSPs.
If the satisfiability problem for the circuit below is decidable, then Goldbach's conjecture can be decided. More precisely, the conjecture fails if and only if there is an assignment for the unassigned input node such that 0 belongs to the set computed by the rightmost node. Note that $\overline{\overline{1} \times \overline{1}} \cap \overline{1} = \mathbb{P}$ where $\mathbb{P}$ is the set of all primes.



Connection of circuits and CSPs. Glaßer et al. [GJM15] combined CSPs and integer circuits, and investigated CSPs over the domain of singletons of natural numbers and with operations from $\{+, \times, \cup, \cap, ^{-}\}$.
As we will see later, each CSP-instance can be represented by a primitive positive fo-sentence such as $\exists X\ (X + X + 4) \cap \big(\mathbb{P} + \mathbb{P}\big) = 0 \cap 1$, where $\mathbb{P}$ can be expressed by $\overline{\overline{1} \times \overline{1}} \cap \overline{1}$. Here we use natural numbers as abbreviations for singletons. Hence, this sentence is true if and only if Goldbach's conjecture does not hold.
However, the set of singletons of natural numbers is not closed under the mentioned operations. As it can be seen in the example above, variables and constants are singletons, whereas there are terms that describe bigger and even infinite sets. Therefore, we consider CSPs over the domain $\mathcal{P}_{\text{fin}}(\mathbb{N}) = \{A \subseteq \mathbb{N} \mid A \text{ is finite}\}$, and replace the complement with the set difference. Thus, compared to the CSPs considered by Glaßer et al. [GJM15] we consider problems that allow a straighter definition (i.e., variables and terms have the same domain), but that more distant to the satisfiability problem for circuits.
Some results of the mentioned papers can be translated to our situation, but in general the questions for the enlarged domain turn out to be different ones.
As soon as one of the arithmetical operations is permitted, the complexity of the CSPs is very high, and it is very difficult to obtain completeness results for certain complexity classes.
Therefore, we also consider a restricted version of the described CSPs, in which the domain is the set of all finite intervals over $\mathbb{N}$, denoted by $[\mathbb{N}]$. Note that this domain is not closed under all permitted operations anymore, but in several cases it is easier to obtain completeness results for this domain.

Our contribution. In the first part we consider all CSPs that only permit set operations. Here, for each problem we are able to show the $\leq_{\text{m}}^{\log}$-completeness for one of the complexity classes L, P, and NP. We observe that there is a case in which the problem over $[\mathbb{N}]$ is less difficult than the corresponding problem over $\mathcal{P}_{\text{fin}}(\mathbb{N})$.
When also admitting arithmetical operations, the complexity is much higher. Each of the problems is $\leq_{\text{m}}^{\log}$-hard for NP, and once both arithmetical operations are permitted, we obtain Σ_1-completeness.
The case where addition is the only operation is particularly interesting. Here for CSPs over arbitrary finite sets we obtain only NP-hardness and membership in NEXP, and one of this paper's open questions is to close this gap. In contrast, the corresponding CSP over $[\mathbb{N}]$ turns out to be to be NP-complete. For the variant over $\mathcal{P}_{\text{fin}}(\mathbb{N})$ we even do not know whether the problem admitting addition and intersection is decidable. The corresponding problem over $[\mathbb{N}]$ is shown to be NP-complete again.
Furthermore, we consider the multiplication of intervals, and show that the CSP over $[\mathbb{N}]$ permitting only multiplication belongs to Σ_3^{p}. This result can be improved if the problem of testing whether two products of intervals are equal can be shown to belong to some class of the polynomial time hierarchy lower than Π_2^{p}.

2 Preliminaries

We start with a couple of short remarks concerning the notations and encodings we use.

2.1 Basic Notations and Encoding of Objects

We use standard notation:
$\mathbb{N}$ is the set of non-negative integers $\{0, 1, 2, \dots\}$, and $\mathbb{N}^+$ is the set of positive integers $\{1, 2, 3, \dots\}$.
Let $\mathcal{P}_{\text{fin}}(\mathbb{N})$ be the set of finite subsets of $\mathbb{N}$. For $A, B \in \mathcal{P}_{\text{fin}}(\mathbb{N})$ we define $A + B =_{\text{def}} \{a + b \mid a \in A, b \in B\}$ and $A \times B =_{\text{def}} \{ab \mid a \in A, b \in B\}$. The set difference is denoted by $-$.
Furthermore, we denote the set of finite intervals over $\mathbb{N}$ by $[\mathbb{N}]$. An interval $\{x \mid a \leq x \leq b\}$ for non-negative integers a and b is represented by $[a, b]$. The set of an interval's endpoints $\{a, b\}$ is denoted by $R([a, b])$. We emphasize that $[a, b] = \emptyset$ in case $a > b$.
Further on let $\{\mathbb{N}\} =_{\text{def}} \{\{n\} \mid n \in \mathbb{N}\}$.

Besides we make use of the following common notations for complexity classes.

complexity class	**contains all problems decidable in**
L	deterministic logarithmic space
NL	non-deterministic logarithmic space
P	deterministic polynomial time
NP	non-deterministic polynomial time
PSPACE	deterministic polynomial space
EXP	deterministic exponential time
NEXP	non-deterministic exponential time

Moreover, for $i \in \mathbb{N}$ we use the common notations Σ_i^{p} and Π_i^{p} for the classes of the polynomial time hierarchy. Σ_1 denotes the set of recursively enumerable sets.

For subsets A and B of $\mathbb{N}$ we denote $A \leq_{\mathrm{m}}^{\log} B$ or $A \leq_{\mathrm{m}}^{\mathrm{p}} B$ if there is a total function $f : \mathbb{N} \to \mathbb{N}$ computable in logarithmic space or polynomial time such that $x \in A \Leftrightarrow f(x) \in B$.

We assume the reader to know that commonly known NP-complete problems like SAT, 3SAT or SOS are even $\leq_{\mathrm{m}}^{\log}$-complete for NP.

Furthermore, in general we assume functions to be total without pointing that out every time. Nevertheless, we will occasionally mention it.
We denote by W_f the set $f(A) = \{f(x) \mid x \in A\}$ for a function $f : A \to B$ and arbitrary sets A and B.

Further on we notice some aspects regarding the encoding of objects by natural numbers or words over some finite alphabet.
We exclusively use standard encodings in this paper. The exact encoding is normally irrelevant. We assume natural numbers to be encoded in binary, dyadic, or k-adic representation for $k > 2$. Anyway, the length of the encoding of a non-negative integer is in $\Theta(\log n)$.
We denote the length of an encoding of an object o by $|o|$. Only for finite sets M the notation $|M|$ refers to the cardinality of M.

We presume that finite subsets of natural numbers $\{a_1, \dots, a_k\}$ for $k \in \mathbb{N}^+$ are encoded such that the encoding's length is in $\Theta(\sum_{i=1}^{k} |a_k|)$. An interval $[a, b]$ for $a < b$ is encoded such that the length of the encoding is in $\Theta(|a| + |b|)$.

We furthermore assume graphs to be represented as adjacency matrices.

At last we remark that we interpret all problems studied in this paper as sets of non-negative integers although for the sake of simplicity we define some problems as sets of sentences or graphs for instance.

2.2 Definition of the Key Problems

We successively define the CSPs this paper deals with. Let X be a variable or a symbol for a constant where symbols for constants are encoding of finite subsets of $\mathbb{N}$. In the latter case we identify the symbol X with the set it encodes.
Then X is a **term**. For terms β and γ as well as $O \subseteq \{+, \times, \cup, \cap, -\}$ and $\oplus \in O$ the expression $(\beta \oplus \gamma)$ is a **term.** $X = Y$ for terms X and Y is an **atom**.
Let $a_1 = (t_{1,1} = t_{1,2}), \dots, a_m = (t_{m,1} = t_{m,2})$ for $m \in \mathbb{N}$ be atoms. Furthermore, let $X_1, \dots, X_n$ be the variables in the mentioned atoms. Then $\varphi = \exists X_1 \dots \exists X_n \; a_1 \wedge \dots \wedge a_m$ is an O**-sentence**, or a **sentence** if O is apparent fromt the context.
Let Var_φ denote the set of variables in φ, and let Const_φ denote the set of constants in φ. Moreover, we denote the set of all terms occurring in φ by T_φ.

We define the semantics of terms. A mapping $\alpha : T_\varphi \to \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ is an **assignment of terms** if the following conditions are satisfied.

- For constants C it holds that $\alpha(C) = C$.
- For all variables X it holds $\alpha(X) \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$.
- For $\oplus \in \{+, \times, \cup, \cap, -\}$ and all terms $X \oplus Y$ it holds that $\alpha(X \oplus Y) = \alpha(X) \oplus \alpha(Y)$.

φ is **true** if and only if there is an assignment of terms α with $\alpha(t_{i,1}) = \alpha(t_{i,2})$ for all $i = 1, \dots, m$. We call such an α **satisfying**.
The restriction of an assignment of terms to Var_φ is called **assignment of variables** or **assignment**. Note that in some cases we also call the restriction of an assignment of terms to $\mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$ **assignment**. We name an assignment of variables β **satisfying** if the assignment of terms induced by β is satisfying. Note that for each assignment of variables β there is exactly one assignment of terms β such that for all $X \in \mathrm{Var}_\varphi$ it holds that $\alpha(X) = \beta(X)$.
With these notions it is possible to define this paper's key problems.

Definition 2.1
Let $O \subseteq \{+, \times, \cup, \cap, -\}$. Then $\mathrm{CSP}\Big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big) =_{\mathrm{def}} \{\varphi \mid \varphi \text{ is a true } O\text{-sentence}\}$.
Furthermore, we define

$$\mathrm{CSP}\Big([\mathbb{N}], \{=\} \cup \mathrm{O}\Big) =_{\mathrm{def}} \{\varphi \mid \varphi \text{ is an } O\text{-sentence, all constants in } \varphi \text{ are intervals,} \\ \text{there is a satisfying assignment } \alpha \text{ with } \alpha(\mathrm{Var}_\varphi) \subseteq [\mathrm{N}]\}.$$

Recall that here all constants are encoded as intervals. That means that a constant $[a, b]$ for $a, b \in \mathbb{N}$ and $a \le b$ is encoded such that the encoding's length is in $\Theta(|a| + |b|)$.

We obtain the following lemma directly from the definition.

Lemma 2.2
For $O \subseteq O' \subseteq \{+, \times, \cup, \cap, -\}$ and $M \in \{\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), [N]\}]$ it holds that

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

We also consider a slightly different problem. This enables us to simplify numerous proofs.

Definition 2.3
Let $O \subseteq \{+, \times, \cup, \cap, -\}$. We define

$$\mathrm{CSP}'\big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \mathrm{O}\big) =_{\mathrm{def}} \{\varphi \mid \varphi \text{ is true, and each atom is of the form } X \oplus Y = Z \\ \text{for } X, Y, Z \in \mathrm{Const}_\varphi \cup \mathrm{Var}_\varphi \text{ and } \oplus \in O\}.$$

Let $\mathrm{CSP}'\big([\mathbb{N}], \{=\} \cup \mathrm{O}\big)$ for $O \subseteq \{\cap, +\}$ be defined analogous to $\mathrm{CSP}'\big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\big)$.

Note that we do not differentiate between the atoms $X \oplus Y = Z$ and $Z = X \oplus Y$ in the definition above.

The following lemma often allows us to presume that an input sentence is of the described simpler form. This is so because we can resolve a bigger term into several smaller terms by storing intermediate results in new variables. For CSPs over intervals, however, this works only when the set $[\mathbb{N}]$ is closed under the permitted operations.

Lemma 2.4
Let $O \subseteq \{+, \times, \cup, \cap, -\}$. *Then* $\mathrm{CSP}'\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big) \equiv_m^{\log} \mathrm{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big)$ *holds.*
If $O \subseteq \{+, \cap\}$ *it also holds that* $\mathrm{CSP}'\Big([\mathbb{N}], \{=\} \cup \mathrm{O}\Big) \equiv_m^{\log} \mathrm{CSP}\Big([\mathbb{N}], \{=\} \cup \mathrm{O}\Big)$.

Proof. We initially prove the first part.
$\leq_{\mathrm{m}}^{\log}$ is trivial. We show the reverse reduction.

Let $\varphi = \exists Z_1 \ldots \exists Z_m \bigwedge_{i=1}^{n} \big(t_{i,1} = t_{i,2}\big)$ be an O-sentence. We sketch the computation of an O-sentence φ' with $\varphi \in \mathrm{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big) \Leftrightarrow \varphi' \in \mathrm{CSP}'\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big)$.

- Let $m_{i,j} \in \mathbb{N}$ be the number of occurrences of symbols from O in the term $t_{i,j}$. For every term $t_{i,j}$ add new variables $X_{i,j,1}, \ldots, X_{i,j,m_{i,j}}$. At the front of the output sentence quantify existentially over all variables $X_{i,j,k_{i,j}}, Z_r$ for $i = 1, \ldots, n$, $j = 1, 2$, $k_{i,j} = 1, \ldots, m_{i,j}$ and $r = 1, \ldots, m$.
- For each term $t_{i,j}$ and all $k = 1, \ldots, m_{i,j}$ do:
  - Let $\oplus_k$ be the kth operation symbol in the term $t_{i,j}$ (counted from left to right).
  - If there is no bracket to the left of $\oplus_k$, but there is a variable or constant directly in front of $\oplus_k$, then let L denote this variable or constant.
  - Otherwise there is at least one closing bracket directly to the left of $\oplus_k$. Find the corresponding opening bracket for that closing bracket. There is a term $\tau \oplus_s \tau'$ for two terms τ, τ' and $s \in \{1, \ldots, m_{i,j}\}$ between these brackets. In that case let L denote the variable $X_{i,j,s}$.
  - Thus, L denotes the term which is the left operand of $\oplus_k$. Proceed analogously on the right side such that R denotes the term which is the right operand of $\oplus_k$.
  - Append the atom $L \oplus_k R = X_{i,j,k}$ to the output sentence.
- For each atom $t_{i,1} = t_{i,2}$ let Y_1 and Y_2 be the variables which stand for $t_{i,1}$ and $t_{i,2}$ respectively. Depending on the availability of operations append $T_1 + \{0\} = T_2$, $T_1 \times \{1\} = T_2$, $T_1 \cup T_1 = T_2$, $T_1 \cap T_1 = T_2$ or $T_1 - \emptyset = T_2$ to the output sentence.

Since at each point in time only a constant number of variables and constants has to be stored and apart from that only the numbers of operation symbols in terms have to be counted, the algorithm is computable in logarithmic space.

Furthermore, the output sentence contains only atoms of the form $X \oplus Y = Z$ for $\oplus \in O$ and variables or constants X, Y, Z.

If φ is true, then φ' is true as well: Let α be a satisfying assignment of terms for φ. Assign $\alpha(Z_1), \ldots, \alpha(Z_m)$ to the variables $Z_1, \ldots, Z_m$ in φ' and extend the mapping constructed that way to an assignment of terms β for φ'. Then β is satisfying as well.

Reversely, if φ' is true, let β be a satisfying assignment of terms for φ'. Now define $\alpha : \mathrm{Var}_{\varphi} \to \mathcal{P}_{\text{fin}}(\mathbb{N}),\ \mathrm{X} \mapsto \beta(\mathrm{X})$. Then α is a satisfying assignment of variables for φ and thus, φ is true.

The second part can be shown analogously. If there is a satisfying assignment of terms α for an O-sentence $\varphi \in \mathrm{CSP}\Big([\mathbb{N}], \{=\} \cup \mathrm{O}\Big)$ with $O \subseteq \{+, \cap\}$, then – as for any two intervals I and J both $I + J \in [\mathbb{N}]$ and $I \cap J \in [\mathbb{N}]$ hold – for each term t it holds $\alpha(t) \in [\mathbb{N}]$. $\square$

Because of the transitivity of $\leq_{\mathrm{m}}^{\log}$ we may hereafter assume for logspace computations and computations with even higher complexity that the input O-sentence is always of the form described in definition 2.3.

The following result shows that for CSPs over $\mathcal{P}_{\text{fin}}(\mathbb{N})$ it is possible to express both "$\cup$" and "$\cap$" by "-".

Lemma 2.5
It holds

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

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

Proof. The statement holds due to lemma 2.4 and because $X = Y \cup Z$ and $X = Y \cap Z$ can also be expressed by $\big((X - Y) - Z = \emptyset\big) \wedge \big(Y - X = \emptyset\big) \wedge \big(Z - X = \emptyset\big)$ and $X = (Y - (Y - X))$. $\square$

Thus, it suffices to consider such problems $\mathrm{CSP}\Big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\Big)$ for $O \subseteq \{\cup, \cap, -, +, \times\}$ for which $- \in O \Rightarrow \{\cup, \cap\} \subseteq O$ holds.

3 CSPs Permitting Only Set Operations

We firstly focus on CSPs permitting no arithmetic but only set operations. This constraint makes the situation essentially easier in two different ways:

On the one hand it will become apparent that all the problems investigated in this section are in NP whereas – as we will see in section 4 – CSPs are NP-hard once an arithmetical operation is permitted, where some of these problems are PSPACE- or even Σ_1-hard.

On the other hand in this section it is much easier to assign CSPs to particular complexity classes. For every subset of $O \subseteq \{\cup, \cap, -\}$ and for $M \in \{\mathcal{P}_{\text{fin}}(\mathbb{N}), [\mathbb{N}]\}$ we prove $\mathrm{CSP}(\mathrm{M}, \{=\} \cup \mathrm{O})$ to be $\leq_{\mathrm{m}}^{\log}$-complete for one of the classes L, P and NP. Thus, all the problems mentioned are considered exhaustively.
This will not be possible in section 4.

3.1 Two Special Cases

We start with the consideration of two special cases. At first we will prove an upper bound for the CSP in which all set operations are admitted. This will be an upper bound for all problems considered in this section. Afterwards we will investigate CSPs in which no operations at all are permitted.

3.1.1 A General Upper Bound

The following lemma shows that all CSPs without arithmetical operation are in NP.

Lemma 3.1
Let $O = \{-, \cup, \cap\}$ and $M \in \{[\mathbb{N}], \mathcal{P}_{\text{fin}}(\mathbb{N})\}$. Then $\mathrm{CSP}\big(\mathrm{M}, \{=\} \cup \mathrm{O}\big) \in \mathrm{NP}$.

Proof. We show the two statements parallel.
Let $\varphi \in \mathrm{CSP}\big(\mathrm{M}, \{=\} \cup \mathrm{O}\big)$. Furthermore, let $K = \bigcup_{C \in \mathrm{Const}_\varphi} C$ in case $M = \mathcal{P}_{\text{fin}}(\mathbb{N})$ and $K = [0, \max\big(\{x \in \bigcup_{C \in \mathrm{Const}_\varphi} C\}\big)]$ in case $M = [\mathbb{N}]$.
Let α be a satisfying assignment of terms for φ. Then β with $\beta(X) = \alpha(X) \cap K$ is a satisfying assignment of terms as well. This can be shown by a structural induction over the definition of an assignment of terms.

Assume β to be non-satisfying. Then there were an atom $t_1 = t_2$ with terms t_1, t_2 such that $\alpha(t_1) = \alpha(t_2)$, but $\alpha(t_1) \cap K \neq \alpha(t_2) \cap K$. Without loss of generality there is an $x \in \big(\alpha(t_1) \cap K\big) - \big(\alpha(t_2) \cap K\big)$. Thus $x \in K$, and hence $x \in \alpha(t_1) - \alpha(t_2)$ which contradicts $\alpha(t_1) = \alpha(t_2)$.

Consequently, whenever there is a satisfying assignment of terms for φ, then there is also a satisfying assignment of terms whose range contains only sets of small numbers, i.e., the occurring numbers are all less than or equal to the greatest number occurring in some constant. Thus, we obtain that the following non-deterministic polynomial time algorithm decides whether $\varphi \in \mathrm{CSP}\big(\mathrm{M}, \{=\} \cup \mathrm{O}\big)$.

1. Guess an assignment of variables with a range $\subseteq \{A \in M \mid A \subseteq K\}$.
2. Test whether the assignment is satisfying, and return 0 or 1 appropriately.

Step 1 can be executed in polynomial time since K only contains $O(|\varphi|)$ elements if $M = \mathcal{P}_{\text{fin}}(\mathbb{N})$. In the other case the interval endpoints have to be chosen from a set with numbers of polynomial length. □

Remark 3.2
When investigating CSPs with exclusively set operations, there occur some NP*-complete problems. The lemma above proves each of them to be in* NP*. Thus, it is sifficient to show the* NP*-hardness for each of these problems. Then the $\leq_{\mathrm{m}}^{\log}$-completeness follows immediately.*

3.1.2 CSPs without Any Operations

Now we consider the case in which there are no operations available at all. $\mathrm{CSP}\big(\mathrm{M}, \{=\}\big)$ for $M \in \{\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), [\mathbb{N}]\}$[1] is the only CSP considered in this paper which we cannot prove to be $\leq_{\mathrm{m}}^{\log}$-hard for P.
We show that this problem is in L. Thereto we use the undirected connectivity problem.

Definition 3.3
Let (V, E) be an undirected graph. Nodes $u, v \in V$ are **connected** if $u = v$, or if there are $k \in \mathbb{N}^+$ and nodes $v_1, v_2, \ldots, v_k$ with $(u, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k), (v_k, v) \in E$.
We define
$\mathrm{USTCON} = \{(V, E, u, v) \mid (V, E)$ is an undirected graph with connected nodes $u, v\}$.

For a long time it was unknown whether USTCON $\in$ L. The directed variant of the problem is known to be NL-complete. Since undirected graphs are specific directed graphs, USTCON $\in$ NL was obvious. Yet there was the hope that the connectivity problem might be easier to decide for the special case of undirected graphs.
Papadimitriou [Pap94] for instance pointed out that the problem is easier than the directed variant because it could be solved in randomized logarithmic space. Nevertheless he was not able to show that the problem belongs to L.

Eventually Reingold [Rei05] proved USTCON to be decidable even in deterministic logarithmic space.

He also indicates that there are log-space computable transformations between natural representations of graphs and thus, “the result is to a large extent independent of the representation of the input graph”.

Therefore, we will assume in the following that graphs are represented by their adjacency matrices.

By use of polynomially many requests to USTCON the following lemma can be shown:

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

Proof. Let $\varphi = \exists X_1 \ldots \exists X_n \, \big(Y_1 = Y_2\big) \wedge \cdots \wedge \big(Y_{2m-1} = Y_{2m}\big)$ for $n, m \in \mathbb{N}$ and constants or variables Y_i be an arbitrary $\emptyset$-sentence.

We consider the undirected graph $G_\varphi = (V_\varphi, E_\varphi)$ with $V_\varphi = \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$ and $E_\varphi = \{\{Y_{2i-1}, Y_{2i}\} \mid i = 1, \ldots, m \wedge Y_{2i-1} \neq Y_{2i}\}$.
The total function f with $\varphi \mapsto G_\varphi$ is in FL as for each atom it only has to be searched the appropriate column and row in the adjacency matrix. This requires only logarithmic space since at each point in time only two references to constants or variables have to be stored[2].

We show

$$\varphi \text{ is not true } \;\Leftrightarrow\; \exists C_1, C_2 \in \mathrm{Const}_\varphi \subseteq \mathrm{V}_\varphi : \big(\mathrm{C}_1 \neq \mathrm{C}_2 \;\wedge\; C_1 \text{ and } C_2 \text{ are connected}\big).$$

“$\Rightarrow$”: We show the contraposition. Let there be no path from C_1 to C_2 for any different constants C_1, C_2. We iteratively construct a satisfying assignment of terms α. Let $\alpha(C) = C$ for each constant C. Furthermore, for every variable X for which there is a $Y \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$ with

- non-defined $\alpha(Y)$ and
- an atom $X = Y$ or $Y = X$ in φ

[1]We assume $M \in \{\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), [\mathbb{N}]\}$ in the following without mentioning that again.
[2]The respective constant cannot necessarily be stored in logarithmic space indeed, but it suffices to store a reference to it. The equality test for two constants of at most linear length can be obviously done in logarithmic space.

let $\alpha(X) = \alpha(Y)$.
After the iterative procedure has become stationary, set $\alpha(X) = \emptyset$ for each remaining variable X. Thus, each variable is mapped onto $\emptyset$ or onto the value of a constant. Consequently, if there was an atom $Y_{2i-1} = Y_{2i}$ not satisfied by φ, then according to the construction of α it would hold that $\alpha(Y_{2i-1}) \neq \emptyset \wedge \alpha(Y_{2i}) \neq \emptyset$. Therefore, we would obtain $\alpha(Y_{2i-1}), \alpha(Y_{2i}) \in \mathrm{Const}_\varphi$. Then, it would follow directly from the construction of α that there is a path[3] from a constant $C = \alpha(Y_{2i-1})$ to Y_{2i-1} and also a path from a constant $C' = \alpha(Y_{2i})$ to Y_{2i}. But then there were a path from C to C' as well, which would be a contradiction due to $C = \alpha(Y_{2i-1}) \neq \alpha(Y_{2i}) = C'$.
"$\Leftarrow$": Connectivity of two nodes $C_1, C_2 \in \mathrm{Const}_\varphi \subseteq \mathrm{V}$ in G_φ means that for every satisfying assignment of terms α, it holds that $\alpha(C_1) = \alpha(C_2)$. Because of $C_1 \neq C_2$ there is no satisfying assignment of terms, and thus, φ is not true.

Consider the following algorithm:

- Input: $\emptyset$-sentence φ
- For any two constants C, C' do:
  - If C and C' are connected in $f(\varphi)$, then return 0.
- Return 1.

It suffices to show that the loop body can be executed in deterministic logarithmic space. This is so because it has to be computed $c_{\mathrm{USTCON}}(f(\varphi), C, C')$ for polynomially many pairs of constants C, C', and each of the values $c_{\mathrm{USTCON}}(f(\varphi), C, C')$ can be computed in logarithmic space. Thus, it follows $\mathrm{CSP}(\mathrm{M}, \emptyset) \in \mathrm{L}$. □

Now it is obvious that $\mathrm{CSP}(\mathrm{M}, \emptyset)$ is $\leq_{\mathrm{m}}^{\log}$-complete for L. Nevertheless, we demonstrate the reduction $\overline{\mathrm{USTCON}} \leq_{\mathrm{m}}^{\log} \mathrm{CSP}(\mathrm{M}, \emptyset)$ to show that a direct proof of $\mathrm{CSP}(\mathrm{M}, \emptyset) \in \mathrm{L}$ is as difficult as a proof for $\mathrm{USTCON} \in \mathrm{L}$.

Let $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ be an undirected graph and $v_i, v_j \in V$. If $i = j$, then return $\emptyset = \{0\}$.
Otherwise return $\varphi = \exists V_1 \ldots \exists V_n \ \big(V_i = \{0\}\big) \wedge \big(V_j = \{1\}\big) \wedge \bigwedge_{(v_s, v_t) \in E} \big(V_s = V_t\big)$.
With a similar reasoning as in the proof of lemma 3.4, it can be shown that there is a path from v_i to v_j if and only if the returned sentence is not true.

3.2 Union

In this section we show the $\leq_{\mathrm{m}}^{\log}$-completeness of $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cup\})$ for P and the $\leq_{\mathrm{m}}^{\log}$-completeness of $\mathrm{CSP}([\mathbb{N}], \{\cup\})$ for NP.

3.2.1 Variant over Arbitrary Finite Sets

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

Proof. According to lemma 2.4 it is sufficient to decide $\mathrm{CSP}'((\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cup\})$ in polynomial time. Hence let $\varphi = \exists X_1 \ldots \exists X_k \ \bigwedge_{i=1}^{t} \big(A_i \cup B_i = C_i\big)$ with $k, t \in \mathbb{N}$ and variables or constants A_i, B_i, C_i.

The following algorithm decides whether there is a satisfying assignment of variables. For this purpose let $K = \bigcup_{C \in \mathrm{Const}_\varphi} C$.

1. Set $\alpha(X) = \begin{cases} X & \text{if } X \text{ is a constant} \\ K & \text{otherwise} \end{cases}$

[3]We use the term path in a way, such that in any graph there is a path from each node to itself, namely the trivial path without any edge.

2. Apply the following rules to every atom $X \cup Y = Z$.
   (a) Set $\alpha(Z) = \alpha(Z) \cap (\alpha(X) \cup \alpha(Y))$.
   (b) Set $\alpha(X) = \alpha(X) \cap \alpha(Z)$ and analogously $\alpha(Y) = \alpha(Y) \cap \alpha(Z)$.
   (c) If $\alpha(C)$ for a constant C has been changed, return 0.
3. If a value $\alpha(X)$ for an $X \in \mathrm{Var}_\varphi$ has been changed in step 2, execute step 2 again.
4. Return 1.

To see that the algorithm works in polynomial time, it suffices to show that step 2 is executed only polynomially often. We obtain this by the fact that for each variable X there are at most $|K|$ changes of $\alpha(X)$.

It remains to show that the algorithm returns 1 if and only if φ is true.

"$\Rightarrow$": Assume that the algorithm returns 1. Let $\alpha(X)$ denote the value of X under α at the end of the algorithm's execution. We now show that α has been constructed such that for each atom $A_i \cup B_i = C_i$ the equation $\alpha(A_i) \cup \alpha(B_i) = \alpha(C_i)$ holds.
Assume that not all equations $\alpha(A_i) \cup \alpha(B_i) = \alpha(C_i)$ are satisfied. Then there is a j such that $\alpha(A_j) \cup \alpha(B_j) \neq \alpha(C_j)$ holds. In the following we write A, B, C for $\alpha(A_j), \alpha(B_j), \alpha(C_j)$ and distinguish the following cases.

- $\exists x \in C - (A \cup B)$: In that case the rule 2a $C = C \cap \big(A \cup B\big)$ would have had to be applied during the last execution of step 2 in which nothing has been changed. Then $x \notin C$ would hold.
- $\exists x \in \big(A \cup B\big) - C$: In that situation the rules of 2b $A = A \cap C$ and $B = B \cap C$ would still have been applied during the last execution of the body loop such that $x \notin A \cup B$ would have held.

Thus, we obtain contradictions in each case. Hence φ is true.

"$\Leftarrow$": Let $\beta : \mathrm{Const}_\varphi \cup \mathrm{Var}_\varphi \to \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ be a satisfying assignment for φ. Thus, $\beta(C) = C$ holds for any $C \in \mathrm{Const}_\varphi$. We may assume that $\beta(\mathrm{Const}_\varphi \cup \mathrm{Var}_\varphi) \subseteq \mathcal{P}(\mathrm{K})$ holds because otherwise $\beta(X)$ could be replaced with $\beta(X) \cap K$, and thus, all equations would still be satisfied.
Before the first execution of step 2

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

obviously holds.
We show: If $(*)$ holds before the application of a rule, then it also holds afterwards.

- Rule 2a: Let U_b denote the value of $\alpha(Z)$ before the application of the rule. Let furthermore U_a denote the value of $\alpha(X)$ after the application of the rule. $\alpha(X)$ and $\alpha(Y)$ denote the respective value at the time of the application of the rule. These values are not changed by applying the rule. If $U_b = U_a$, then $(*)$ also holds after applying the rule. Otherwise there is an $x \in U_b - U_a$. Hence $x \notin \alpha(X) \cup \alpha(Y)$. Then due to $(*)$ it follows $x \notin \beta(X) \cup \beta(Y)$. Thus, $x \notin \beta(Z)$ holds. Therefore, $(*)$ also holds after the application of the rule.
- Rule 2b: Obviously it is enough to consider only the first of the two equations. Let U_b denote the value of $\alpha(X)$ before applying the rule and U_a the value after the application. $\alpha(Z)$ refers to the value at the time of the application of the rule. This value is not changed by applying the rule. If $U_b = U_a$, then $(*)$ also holds after applying the rule. Otherwise there is an $x \in U_b - U_a$. Thus $x \notin \alpha(Z)$. Then $(*)$ follows due to $x \notin \beta(Z)$, and by that $x \notin \beta(X)$ as well. Therefore, $(*)$ also holds after the application of the rule.

Particularly, for any constant C and at any point in time during the execution of the algorithm $C = \beta(C) \subseteq \alpha(C) \subseteq C$ holds. Hence $\alpha(C)$ is not changed, and thus, the algorithm returns 1. $\square$

$\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cup\})$ is even $\leq_{\mathrm{m}}^{\mathrm{log}}$-hard for P. This is shown by the proof of the following lemma.

Lemma 3.6
$\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cup\})$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* P.

Proof. We make use of the problem MONOTONE CIRCUIT VALUE EVALUATION, which was shown to be $\leq_{\mathrm{m}}^{\log}$-hard for P by Greenlaw et al. [GHR91]. We denote this problem by MCVE.
Consider the following algorithm:

- Input: a monotone boolean circuit α with input gates $x_1, \dots, x_n \in \{0,1\}$. Let the OR-gates be denoted by $o_1, \dots, o_k$ and the AND-gates by $a_1, \dots, a_m$ such that without loss of generality o_k denotes the output gate.

- We successively desribe a sentence φ. Let

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

  Hence, there is one variable for each OR- and each AND-gate. Furthermore, there are two more auxiliary variables for each AND-gate.

- Set $\psi(X_1, \dots, X_n, O_1, \dots, O_k, A_1, \dots, A_m, H_1, H_1' \dots, H_m, H_m')$ to

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

  In the following the set $\{1\}$ stands for the boolean value 1 in the boolean circuit whereas $\emptyset$ stands for 0.

- For an OR-gate o_i and the two variables A, B, that stand for the inputs of o_i, set $\chi_{o_i} = (O_i = A \cup B)$.

- Analogously, for an AND-GATE a_i and variables A, B which stand for the inputs of a_i set $\chi_{a_i} = (A_i \cup H_i = A) \wedge (A_i \cup H_i' = B)$.

- Return $f(\alpha) = \varphi$.

Let f be the function computed by the algorithm.

Since at any point in time only constantly many variables or constants have to be stored, f can be computed in logarithmic space. Furthermore, it can be easily seen that for a circuit α which returns 1 on input $x_1, \dots, x_n$ the sentence $f(\alpha)$ is true: to obtain a satisfying assignment of variables, it suffices to assign the value which stands for the ouput value of a gate o_i or a_j to the corresponding variable O_i or A_j. For OR-gates this is required by the sentence. For AND-gates it is possible to choose an assignment which satisfies all the sentence's atoms. It remains to show the reverse statement:

Let $\varphi = f(\alpha)$ for monotone circuit α with inputs $x_1, \dots, x_n$ be a true sentence. We show that then the circuit α returns 1 on input $x_1, \dots, x_n$.
As φ is true, there is a mapping

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

such that

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

is true. Then $W_\beta \subseteq \{\{1\}, \emptyset\}$ holds:
This is so because the $\beta(X_i)$ must be equal to $\{1\}$ or $\emptyset$ according to the sentence, and all $\beta(O_i)$, $\beta(A_j)$, $\beta(H_j)$, $\beta(H_j')$ have to be a subset of the union of the two "incoming variables".
We consider the circuit α on input $x_1, \dots, x_n$. Let $\gamma : \{o_1, \dots, o_k, a_1, \dots, a_m\} \to \{0,1\}$ be the function which maps every gate onto its output value.
We show the following statements inductively. Assume $\max(\emptyset) = 0$.

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

The statements in (A) are obvious. Now, let g be a gate, and let G be the corresponding variable. The gates's inputs are inputs of the circuit, OR-, or AND-gates. Assume that for their values a, b and the corresponding variables A, B the following holds

$$\max(\beta(A)) \leq a \quad \text{and} \quad \max(\beta(B)) \leq b.$$

We distinguish two cases:

- g is an OR-gate. In case $\gamma(g) = 1$, trivially the condition holds. Otherwise we have $a = b = 0$, and thus $\beta(A) = \emptyset$ and $\beta(B) = \emptyset$, by which we also obtain $\beta(G) = \beta(A) \cup \beta(B) = \emptyset$. This shows (B).
- g is an AND-gate. If $\gamma(g) = 1$, the statement (C) is obviously true. Otherwise without loss of generality there is an a with $a = 0$, and thus $\beta(A) = \emptyset$. Due to $\beta(G) \subseteq \beta(A)$ we obtain $\beta(G) = \emptyset$. This shows (C).

From $1 = \max(\beta(O_k)) \leq \gamma(o_k) \leq 1$ it follows $\gamma(o_k) = 1$. Thus the proof is complete. □

Now the following theorem follows directly.

Theorem 3.7
$\mathrm{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{\cup\})$ *is* $\leq_{\text{m}}^{\text{log}}$*-complete for* P.

Proof. The statement can be deduced from lemma 3.5 and lemma 3.6. □

3.2.2 Variant over Finite Intervals

Assume P $\neq$ NP. Then, contrary to expectation the problem $\mathrm{CSP}([\mathbb{N}], \{=, \cup\})$ is harder than the problem $\mathrm{CSP}([\mathcal{P}_{\text{fin}}(\mathbb{N})], \{=, \cup\})$. Since the variant over $[\mathbb{N}]$ might firstly appear to be a restriction of the variant over $\mathcal{P}_{\text{fin}}(\mathbb{N})$, one would at first view rather expect the opposite. The $\{\cup\}$-sentences over $[\mathbb{N}]$ are indeed a restricted version of sentences over $\mathcal{P}_{\text{fin}}(\mathbb{N})$ because the only difference is that the constants have to be specific elements of $\mathcal{P}_{\text{fin}}(\mathbb{N})$, namely finite intervals.
As a result of the fact that in the variant over intervals also the variables must be mapped onto intervals, we obtain greater expressive power in this specific situation.
For instance, the sentence $\exists X \exists Y \; \{0\} \cup \{2\} = X \cup Y$ expresses that exactly one of the two variables has to be mapped onto $\{0\}$ under a satisfying assignment of variables whereas the value $\{2\}$ has to be assigned to the other one.
It is not possible to express this once arbitrary finite subsets can be assigned to variables.

With the help of such tricks we are able to show 3SAT $\leq_{\text{m}}^{\text{log}} \mathrm{CSP}([\mathbb{N}], \{\cup\})$.

Lemma 3.8
3SAT $\leq_{\text{m}}^{\text{log}} \mathrm{CSP}([\mathbb{N}], \{\cup\})$.

Proof. Let H be a propositional formula in conjunctive normal form with exactly three literals per clause and variables $x_1, \ldots, x_n$. That means $H = \bigwedge_{i=1}^{m} K_i$ for clauses $K_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ with literals $l_{i,j} \in \{x_1, \neg x_1, x_2, \neg x_2, \ldots, x_n, \neg x_n\}$.

We sketch a $\{\cup\}$-sentence

$$\varphi = \exists X_1 \ldots \exists X_n \exists X_1' \ldots \exists X_n' \exists L_{i,1} \exists L_{i,2} \exists L_{i,3} \ldots \exists L_{m,1} \exists L_{m,2} \exists L_{m,3} \;\; \psi_1 \wedge \psi_2 \wedge \psi_3$$

with $\varphi \in \mathrm{CSP}([\mathbb{N}], \{\cup\}) \Leftrightarrow$ H $\in$ 3SAT.
Thus, for each propositional variable there are two variables in φ. These variables' purpose is to describe the truth value of the corresponding propositional variable and its negation. Let $\{2\}$ stand for the truth value 1, and let $\{0\}$ stand for the truth value 0. Set $\psi_1 = \bigwedge_{r=1}^{n} \big(X_r \cup X_r' = \{0\} \cup \{2\}\big)$.

Hence, under a satisfying assignment X_r and X'_r are mapped onto either $\{0\}$ or $\{2\}$, and X_r is mapped onto $\{0\}$ if and only if X'_r is mapped onto $\{2\}$. Thus, X_r describes the value of x_r, and X'_r describes the value of $\neg x'_r$.
Let $s : \{(i,j) \mid i \in \{1,\dots,m\}, j \in \{1,2,3\}\} \to \{1,\dots,n\}$ map the pair (i,j) onto the index of the variable in $l_{i,j}$. Now it has to be made sure that each $L_{i,j}$ is mapped onto the correct value of $\{X_{s(i,j)}, X'_{s(i,j)}\}$. We require this in φ by

$$\psi_2 = \bigwedge_{i=1}^{m} \bigwedge_{j=1}^{3} \left(L_{i,j} = \begin{cases} X_{s(i,j)} & \text{if } l_{i,j} = x_{i,j} \\ X'_{s(i,j)} & \text{if } l_{i,j} = \neg x_{i,j} \end{cases}\right).$$

If there is an assignment for the propositional variables of ψ such that for each clause, there is at least one literal with the truth value 1, ψ is satisfiable. This can be described in φ as follows:

$$\psi_3 = \bigwedge_{1 \le i \le m} \Big(L_{i,1} \cup L_{i,2} \cup L_{i,3} \cup \{0\} = \{0\} \cup \{2\}\Big).$$

Hence $\varphi \in \mathrm{CSP}([\mathbb{N}], \{\cup\}) \Leftrightarrow \mathrm{H} \in 3\mathrm{SAT}$.
With the observation that at any point in time of the algorithm's execution only constantly many indices of the polynomially many variables have to be stored, and thus, only logarithmic space is required for the computation of the reduction, the proof is complete. □

Now we immediately obtain.

Theorem 3.9
$\mathrm{CSP}([\mathbb{N}], \{\cup\})$ *is* $\le_{\mathrm{m}}^{\log}$*-complete for* NP.

Proof. The hardness follows from lemma 3.8. $\mathrm{CSP}([\mathbb{N}], \{\cup\})$ is in NP according to remark 3.2. □

3.3 Intersection

Analogously to section 3.2.1 we show $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cap\})$ to be $\le_{\mathrm{m}}^{\log}$-complete for P. Unlike for $\mathrm{CSP}([\mathbb{N}], \{\cup\})$ we cannot prove the variant over finite intervals to be NP-complete[4]. Instead we will see that $\mathrm{CSP}([\mathbb{N}], \{\cap\})$ is $\le_{\mathrm{m}}^{\log}$-complete for P.

3.3.1 Variant over Arbitrary Finite Sets

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

Proof. Like in the proof of lemma 3.5 and according to lemma 2.4 it suffices to show that there is a polynomial time algorithm which decides whether a sentence $\varphi = \exists X_1 \dots \exists X_k \bigwedge_{i=1}^{t} \big(A_i \cap B_i = C_i\big)$ with $k, t \in \mathbb{N}$ and variables or constants A_i, B_i, C_i belongs to $\mathrm{CSP}'(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cap\})$.

The following algorithm decides whether there is a total $\alpha : \mathrm{Const}_\varphi \cup \mathrm{Var}_\varphi \to \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ with $\alpha(C) = C$ for all constants C such that the equations $\alpha(A_i) \cap \alpha(B_i) = \alpha(C_i)$ hold.

1. Set $\alpha(X) = \begin{cases} X & \text{if } X \text{ constant} \\ \emptyset & \text{otherwise} \end{cases}$.
2. For each atom $X \cap Y = Z$ apply the following rules.
   (a) Set $\alpha(Z) = \alpha(Z) \cup (\alpha(X) \cap \alpha(Y))$.
   (b) Set $\alpha(X) = \alpha(X) \cup \alpha(Z)$ and analogously $\alpha(Y) = \alpha(Y) \cup \alpha(Z)$.
   (c) If $\alpha(C)$ for a constant C has been changed, return 0.
3. If $\alpha(X)$ for an $X \in \mathrm{Var}_\varphi$ has been changed in step 2, then execute step 2 again.

[4]Probably this cannot be proven at all.

4. Return 1.

Similarly to the proof of lemma 3.5 it now can be shown that the algorithm returns 1 if and only if φ is true. □

By a reduction similar to the reduction in the proof of lemma 3.6 the $\leq_{\mathrm{m}}^{\log}$-hardness of the problem $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cap\})$ for P can be shown.

Lemma 3.11
$\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cap\})$ and $\mathrm{CSP}([\mathbb{N}], \{\cap\})$ are $\leq_{\mathrm{m}}^{\log}$-hard for P.

Proof. We show both statements parallel. The following algorithm computes a reduction from MCVE to the two mentioned problems.

- Input: a monotone boolean circuit α with inputs $x_1, \ldots, x_n \in \{0,1\}$. Let the OR-gates be denoted by $o_1, \ldots, o_k$ and the AND-gates by $a_1, \ldots, a_m$. Without loss of generality o_k denotes the ouput gate.
- We successively build a sentence φ. For that purpose let
$$\begin{aligned}\varphi = &\exists X_1 \ldots \exists X_n \exists O_1 \ldots \exists O_k \exists A_1 \ldots \exists A_m \exists H_1 \exists H_1' \ldots \exists H_m \exists H_m' \\ &\psi(X_1, \ldots, X_n, O_1, \ldots, O_k, A_1, \ldots, A_m, H_1, H_1' \ldots, H_m, H_m').\end{aligned}$$
  Hence, for each input gate and each OR- and AND-gate, there is a corresponding variable and for each AND-gate, there are two more auxiliary variables.
- Set
$$\begin{aligned}&\psi(X_1, \ldots, X_n, O_1, \ldots, O_k, A_1, \ldots, A_m, H_1, H_1' \ldots, H_m, H_m') = \\ &\bigwedge_{i=1}^{n} \left(X_i = \begin{cases} \{1\} & \text{if } x_i = 0 \\ \emptyset & \text{otherwise} \end{cases} \right) \wedge \bigwedge_{i=1}^{k} \chi_{o_i} \wedge \bigwedge_{i=1}^{m} \chi_{a_i} \wedge (O_k = \emptyset).\end{aligned}$$
- If o_i is an OR-gate and A, B are the variables which stand for the inputs of o_i, set $\chi_{o_i} = (O_i = A \cap B)$.
- If a_i is an AND-gate and A, B are the variables that stand for the inputs of a_i, set $\chi_{a_i} = \big(A_i \cap H_i = A\big) \wedge \big(A_i \cap H_i' = B\big)$.
- Return $f(\alpha) = \varphi$.

Let f denote the function computed by the algorithm. f can be computed in logarithmic space since at each point in time, there are only constantly many variables or constants that have to be stored.

It remains to show
$$\alpha \in \mathrm{MCVE} \Leftrightarrow f(\alpha) = \varphi \in \mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cap\}) \Leftrightarrow \mathrm{f}(\alpha) \in \mathrm{CSP}([\mathbb{N}], \{\cap\}).$$

The right equivalence obviously holds. The left equivalence can be proven analogously to the proof of lemma 3.6. □

It follows directly:

Theorem 3.12
$\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cap\})$ is $\leq_{\mathrm{m}}^{\log}$-complete for P.

Proof. The statement follows from lemma 3.10 and lemma 3.11. □

3.3.2 Variant over Finite Intervals

Also $\mathrm{CSP}([\mathbb{N}], \{\cap\})$ can be shown to belong to P.

Lemma 3.13
$\mathrm{CSP}([\mathbb{N}], \{\cap\}) \in \mathrm{P}$.

Proof. According to lemma 2.4 it suffices to show $\mathrm{CSP}'\big([\mathbb{N}], \{=, \cap\}\big) \in \mathrm{P}$. Hence, let φ be a $\{\cap\}$-sentence whose atoms are all of the form $A \cap B = C$.
We present an iterative algorithm, which decides $\mathrm{CSP}([\mathbb{N}], \{\cap\})$. Consider the mapping $\alpha : \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi \to [\mathbb{N}]$ defined by $\alpha(X) = \emptyset$ for variables X and $\alpha(C) = C$ for $C \in \mathrm{Const}_\varphi$. The following algorithm changes this mapping's values:

1. For each atom $X \cap Y = Z$ apply the following rules:
   (a) First set $\alpha(Z) = \alpha(Z) \cup (\alpha(X) \cap \alpha(Y))$. Afterwards set
   $$\alpha(Z) = [\min\big(\alpha(Z)\big), \max\big(\alpha(Z)\big)].$$
   (b) Set $\alpha(X) = \alpha(X) \cup \alpha(Z)$ and then $\alpha(X) = [\min\big(\alpha(X)\big), \max\big(\alpha(X)\big).]$ Proceed analogously for Y.
   (c) If $\alpha(C)$ for a constant C has been changed, return 0.
2. If there has been a change of a value $\alpha(X)$ for an $X \in \mathrm{Var}_\varphi$ in step 1, execute 1 again.
3. Return 1.

In each execution of step 1 except for the last one there is at least one change of a value $\alpha(X)$ for a variable X.
It can be shown inductively that at each point in time during the execution of the algorithm, it holds for each variable X that $\alpha(X) = [a, b]$, where $[a, b] = \emptyset$ or both a and b are equal to some lower or upper interval endpoint of a constant in φ[5]:
The induction basis is trivial. Assume that now $\alpha(X)$ for an $X \in \mathrm{Var}_\varphi$ is changed. We distinguish two cases:

1. By applying rule 1a to $X = Y \cap Z$ for $Y, Z \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$ there is a change of $\alpha(X)$. According to the induction hypothesis before the application of the rule the endpoints of $\alpha(X), \alpha(Y), \alpha(Z)$ also occur as endpoints of constants. Then this holds for $\alpha(Y) \cap \alpha(Z)$ as well, and hence also for $\alpha(X)$ after the application of the rule.
2. For the case $Z = X \cap Y$ for $Y, Z \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$ it can be argued analogously.

Since there are only linearly many interval endpoints of constants, and the values $\alpha(X)$ for variables X change monotonically[6], there are only polynomially many executions of step 1. Thus, the algorithm only requires polynomial time.

We now show that the algorithm returns 1 if and only if there is a satisfying assignment of the variables in φ.

"$\Rightarrow$": We consider the case in which the algorithm returns 1 and show that then $\alpha_{|\mathrm{Var}_\varphi}$[7] is a satisfying assignment. Assume that this is not so. Then there is an atom $Z = X \cap Y$ with $\alpha(X) \cap \alpha(Y) \neq \alpha(Z)$. We distinguish two cases.

1. $\exists x \in \big(\alpha(X) \cap \alpha(Y)\big) - \alpha(Z)$. In that situation the algorithm would not have terminated returning 1 but applied the rule 1a to $\alpha(Z)$.

[5]Assume for the rest of the section that there are two different numbers occurring as endpoints of some constant. Note that then, for $[a, b] = \emptyset$ we can choose a, b from the set of interval endpoints of constants such that $a > b$ and thus, also in that case the latter condition holds.
[6]This means that the lower endpoint of X decreases monotonically whereas the upper endpoint increases monotonically.
[7]The mapping $\alpha_{|\mathrm{Var}_\varphi}$ changes during the execution of the algorithm. Here, it is referred to the function after the execution of the algorithm.

2. $\exists x \in \alpha(Z) - \big(\alpha(X) \cap \alpha(Y)\big)$. Here instead of terminating the algorithm would have applied the rule 1b.

Consequently α is a satisfying assignment of variables.

"$\Leftarrow$": Let $\beta : \mathrm{Var}_\varphi \to [\mathbb{N}]$ be a satisfying assignment of variables for φ. We farther write $\beta(C)$ for the value of a constant C, and show that at any point in time $\alpha(X) \subseteq \beta(X)$ for all $X \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi$. At the beginning of the algorithm's execution this is obviously true. For an arbitrary atom $X \cap Y = Z$ we write $U =_{\mathrm{def}} \alpha(X)$, $V =_{\mathrm{def}} \alpha(Y)$, and $W =_{\mathrm{def}} \alpha(Z)$ for the current values. Now we consider what happens when one of the rules is applied, and write U', V', and W' for the values of $\alpha(X)$, $\alpha(Y)$, and $\alpha(Z)$ after the application of the respective rule.

1. Consider the application of rule 1a. Then $U \subseteq \alpha(X)$, $V \subseteq \alpha(Y)$, and $W \subseteq \alpha(Z)$. Furthermore, it holds $W' = W \cup (U \cap V) \subseteq \beta(Z) \cup \big(\beta(X) \cap \beta(Y)\big) \subseteq \beta(Z)$.
2. Now consider the application of rule 1b. We only discuss the application for X. Then $U \subseteq \alpha(X)$ and $W \subseteq \alpha(Z)$. Due to $\beta(Z) \subseteq \beta(X)$ we obtain $U' = U \cup W \subseteq \beta(X) \cup \beta(Z) \subseteq \beta(X)$.

In particular for each constant C, it holds $\alpha(C) \subseteq \beta(C)$ at any point in time wherefore the algorithm does not return 0. As described above, the algorithm terminates after polynomially many steps of computation, and returns 1. $\square$

Theorem 3.14
$\mathrm{CSP}([\mathbb{N}], \{\cap\})$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* P.

Proof. We obtain the statement from lemma 3.13 and lemma 3.11. $\square$

3.4 Intersection and Union

In this section we prove the NP-completeness of $\mathrm{CSP}\big(\mathrm{M}, \{=, \cup, \cap\}\big)$ for $M = [\mathbb{N}]$ and $M = \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$.

3.4.1 Variant over Arbitrary Finite Sets

We show the $\leq_{\mathrm{m}}^{\log}$-hardness for NP by a reduction from 3SAT.

Lemma 3.15
$\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cup, \cap\})$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* NP.

Proof. We show $\mathrm{3SAT} \leq_{\mathrm{m}}^{\log} \mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cup, \cap\})$. As 3SAT is $\leq_{\mathrm{m}}^{\log}$-complete for NP, this is sufficient. Let ψ be an arbitrary propositional formula in conjunctive normal form with exactly 3 literals per clause and variables $x_1, \ldots, x_n$, hence $\psi = \bigwedge_{i=1}^{r}(l_{i,1} \vee l_{i,2} \vee l_{i,3})$ for literals $l_{i,j} \in \{x_{i,j}, \neg x_{i,j}\}$ and $x_{i,j} \in \{x_1, \ldots, x_n\}$.

We construct a sentence φ such that $\varphi \in \mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\cup, \cap\}) \Leftrightarrow \psi \in \mathrm{3SAT}$. Here the set $\{0\}$ describes the truth value 0, and the set $\{0, 1\}$ stands for the truth value 1.

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

The second line of the sentence expresses that for $i = 1, \ldots, n$ and each satisfying assignment α, it holds $\alpha(X_i) \in \{\{0\}, \{0,1\}\}$.

The third line requires that the variables $L_{i,j}$, which stand for the literals $l_{i,j} = \neg x_s$, must be mapped onto an element of $\{\{0,1\}, \{0\}\}$ by a satisfying assignment of variables α, and that $\alpha(L_{i,j}) = \{0,1\}$ if and only if $\alpha(X_s) \neq \{0,1\}$.

In the fourth line we express that for the variable $L_{i,j}$, which stands for a literal $l_{i,j} = x_s$, the equation $\alpha(X_s) = \alpha(L_{i,j})$ holds for any satisfying assignment of variables α.

f can be computed in logarithmic space because at any point in time of the computation it suffices to store constantly many variables and the number of variables and clauses in ψ.

For an assignment γ of the propositional variables x_i we define an assignment of variables α for φ as follows:
If $\gamma(x_i) = 1$, set $\alpha(X_i) = \{0, 1\}$. Otherwise set $\alpha(X_i) = \{0\}$. Then γ is satisfying for ψ if and only if α is satisfying for φ. $\square$

As a result the following theorem can be proven:

Theorem 3.16
$\mathrm{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{\cup, \cap\})$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* NP.

Proof. The statement is implied by lemma 3.4 and remark 3.2. $\square$

3.4.2 Variant over Finite Intervals

With the help of lemma 3.8 the following theorem can be shown:

Theorem 3.17
$\mathrm{CSP}([\mathbb{N}], \{\cup, \cap\})$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* NP.

Proof. Lemma 3.8 and lemma 2.2 imply the hardness of $\mathrm{CSP}([\mathbb{N}], \{\cup, \cap\})$ for NP. We obtain $\mathrm{CSP}([\mathbb{N}], \{\cup, \cap\}) \in \mathrm{NP}$ by remark 3.2. $\square$

3.5 Set Difference

This section's purpose is the investigation of CSPs which permit the set difference and an arbitrary subset of other set operations.

When considering CSPs over $\mathcal{P}_{\text{fin}}(\mathbb{N})$, the situation is quite simple as the following remark underlines.

Remark 3.18
Consider $\mathrm{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\big)$ *for* $O \subseteq \{-, \cap, \cup\}$ *with* $- \in O$*. According to remark 3.2 this problem belongs to* NP*. According to lemma 3.15* $\mathrm{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{\cup, \cap\})$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* NP*. Hence, Lemma 2.2 yields the* $\leq_{\mathrm{m}}^{\log}$*-hardness of* $\mathrm{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=, \cup, \cap, -\}\big)$*. Consequently we obtain the* $\leq_{\mathrm{m}}^{\log}$*-completeness of* $\mathrm{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\big)$ *for arbitrary* $O \subseteq \{-, \cap, \cup\}$ *with* $- \in O$ *for* NP *by lemma 2.5.*

The situation for CSPs over $[\mathbb{N}]$ is a bit more complicated. From lemma 3.2 we again obtain NP as an upper bound. Yet also the $\leq_{\mathrm{m}}^{\log}$-hardness for NP of

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

can be shown. Our approach is similar to lemma 3.8.

Theorem 3.19
$\mathrm{CSP}\big([\mathbb{N}], \{=\} \cup \mathrm{O}\big)$ *for* $O \subseteq \{-, \cap, \cup\}$ *with* $- \in O$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* NP.

Proof. As remarked above it is sufficient to show the $\leq_{\mathrm{m}}^{\log}$-hardness for NP. Due to lemma 2.2 it suffices to prove $3\mathrm{SAT} \leq_{\mathrm{m}}^{\log} \mathrm{CSP}\big([\mathbb{N}], \{=, -\}\big)$.

Let H be a propositional formula in conjunctive normal form with exactly 3 variables per clause and variables $x_1, \dots, x_n$, hence $H = \bigwedge_{i=1}^{m} \big(\bigvee_{j=1}^{3} l_{i,j}\big)$ for literals $l_{i,j}$. Furthermore, let $s : \{(i,j) \mid i \in \{1, \dots, m\}, j \in \{1,2,3\}\} \to \{1, \dots, n\}$ be a function such that $l_{i,j} \in \{x_{s(i,j)}, \neg x_{s(i,j)}\}$.

We describe a sentence φ which is true if and only if H is satisfiable. Here the truth value 0 is represented by the set $[0]$ and the truth value 1 by the set $[2]$. Therefore, our sentence expresses that the CSP-variables X_i which stand for the variables in H must be mapped onto an element of

$\{[0],[2]\}$ by a satisfying assignment. Note that X'_i is mapped onto the respective other value (line 2). Further on we use variables $L_{i,j}$ which stand for the literals.
Finally (line 4) it must be required that for all $i = 1,\dots,m$ there is at least one literal $L_{i,j}$ in the i-th clause which is mapped onto the value [2].

$$\exists X_1 \exists X'_1 \dots \exists X_n \exists X'_n \exists L_{1,1} \exists L_{1,2} \exists L_{1,3} \dots \exists L_{m,1} \exists L_{m,2} \exists L_{m,3}$$
$$\bigwedge_{i=1,\dots,n} \Big(\big([0,2] - X\big) - X' = \emptyset \wedge X - ([0,2] - [1]) = \emptyset \wedge X' - ([0,2]-[1]) = \emptyset \Big)$$
$$\bigwedge_{j=1,\dots,m} \bigwedge_{k=1}^{3} L_{j,k} = \begin{cases} X_{s(j,k)} & \text{if } l_{j,k} = x_{s(j,k)} \\ X'_{s(j,k)} & \text{if } l_{j,k} = \neg x_{s(j,k)} \end{cases}$$
$$\bigwedge_{j=1,\dots,m} \Big(\big([2] - L_{i,1}\big) - L_{i,2} \Big) - L_{i,3} = \emptyset.$$

□

3.6 Overview and Conclusion

The following tables provide an overview over the results obtained in this section. The first table deals with the CSPs over $\mathcal{P}_{\text{fin}}(\mathbb{N})$.

$\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \text{O}\big)$ with $O =$	$\leq_m^{\log}$-hard for	member of
$\emptyset$	L	L, 3.4
$\{\cup\}$	P, 3.2	P, 3.2
$\{\cap\}$	P, 3.3	P, 3.3
$\{\cup,\cap\}$	NP, 3.4	NP, 3.2
$\{-\}$	NP, 2.5	NP, 3.2
$\{-,\cup\}$	NP, 2.5	NP, 3.2
$\{-,\cap\}$	NP, 2.5	NP, 3.2
$\{-,\cup,\cap\}$	NP, 2.5	NP, 3.2

The succeeding table gives information about the CSPs over $[\mathbb{N}]$.

$\text{CSP}\big(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=\} \cup \text{O}\big)$ with $O =$	$\leq_m^{\log}$-hard for	member of
$\emptyset$	L	L, 3.4
$\{\cup\}$	NP, 3.8	NP, 3.2
$\{\cap\}$	P, 3.3	P, 3.3
$\{\cup,\cap\}$	NP, 3.4	NP, 3.4
$\{-\}$	NP, 2.5	NP, 3.2
$\{-,\cup\}$	NP, 2.5	NP, 3.2
$\{-,\cap\}$	NP, 2.5	NP, 3.2
$\{-,\cup,\cap\}$	NP, 2.5	NP, 3.2

Thus, for each of the problems the $\leq_m^{\log}$-completeness for one of the complexity classes L, P, NP is proven.
Further on it can be seen that in both situations we receive the same results for almost all cases. The only exception is $\text{CSP}\big(\text{M}, \{=,\cup\}\big)$ for $M \in \{[\mathbb{N}], \mathcal{P}_{\text{fin}}(\mathbb{N})\}$. This is the only case throughout this paper where – under the assumption P $\neq$ NP – deciding a problem over $[\mathbb{N}]$ is more difficult than deciding the corresponding problem over $\mathcal{P}_{\text{fin}}(\mathbb{N})$. In section 4 there are some examples for which under the assumption NP $\neq$ PSPACE the reverse statement holds.

4 CSPs Permitting Arithmetic Operations

Before we move to the consideration of some specific problems, we show an upper bound for all CSPs investigated in this paper:

Theorem 4.1
Let $M \in \{\mathcal{P}_{\text{fin}}(\mathbb{N}), [\mathbb{N}]\}$ and $O \subseteq \{+, \times, \cup, \cap, -\}$. Then $\mathrm{CSP(M, O)} \in \Sigma_1$.

Proof. The set

$$\{(\varphi, \alpha) \mid \varphi \in \mathrm{CSP(M, O)},\ \alpha \text{ is a satisfying assignment of variables for } \varphi\}$$

is decidable. Hence, $\mathrm{CSP(M, O)}$ is a projection of a decidable set, and thus $\mathrm{CSP(M, O)} \in \Sigma_1$. □

4.1 Addition

In this section we show that $\mathrm{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+\})$ is NP-hard and belongs to NEXP, and prove $\mathrm{CSP}([\mathbb{N}], \{+\})$ to be NP-complete.

At first we show: if there is a satisfying assignment of variables for a given sentence φ, then there is also a satisfying assignment for φ satisfying some upper bound.

Lemma 4.2
Let $\varphi \in \mathrm{CSP'}\Big(\mathrm{M}, \{=, +\}\Big)$ for $M \in \{\mathcal{P}_{\text{fin}}(\mathbb{N}), [\mathbb{N}]\}$. Further on let $x = \max\{\bigcup_{C \in \mathrm{Const}_\varphi} C\}$[8] *and $n = |\varphi|$. Then there is a satisfying assignment α for φ with*

$$\forall X \in \mathrm{Var}_\varphi : \quad \max(\alpha(\mathrm{X})) \leq \mathrm{x2^n}.$$

Proof. The following non-deterministic algorithm successively constructs an initially undefined assignment α.

1. Set $\alpha(C) = C$ for each constant C.
2. For each variable X with undefined $\alpha(X)$ occurring in an atom $X = Y + Z$ such that $\alpha(Y)$ and $\alpha(Z)$ are already defined, set $\alpha(X) =_{\text{def}} \alpha(Y) + \alpha(Z)$.
3. For each variable X with undefined $\alpha(X)$ occurring in an atom $X + Z_1 = Z_2$ such that $\alpha(Z_2)$ is defined and unequal to $\emptyset$, guess a set $S \in M$ with $\max\{S\} \leq \max(\alpha(Z_2))$, and set $\alpha(X) =_{\text{def}} S$.
4. If there is a variable X such that $\alpha(X)$ has been defined in the last execution of the steps 2 and 3, then go to step 2.
5. For all variables X with undefined $\alpha(X)$ define $\alpha(X) =_{\text{def}} \emptyset$.
6. If α is satisfying, then return α.

If the algorithm returns an assignment on one computation path, then this assignment is obviously satisfying.
Thus, it suffices to show that the algorithm returns an assignment on at least one computation path. The algorithm terminates on each computation path since there is one more loop iteration only if there is a variable whose value has been changed in the last iteration. Thus, there are at most $|\mathrm{Var}_\varphi|$ loop iterations.

Let β be a satisfying assignment for φ. We show inductively that before every loop iteration there is a computation path such that for the assignment α constructed on that path the following holds:

$$\forall X \in \mathrm{Var}_\varphi : \ \alpha(\mathrm{X}) \text{ non-defined } \vee \alpha(\mathrm{X}) = \beta(\mathrm{X}).$$

Before the first loop iteration the condition holds obviously. Assume the condition is satisfied before an arbitrary loop iteration.

[8] Let $\max(\emptyset) =_{\text{def}} -\infty$ and $-\infty \leq \zeta$ for $\zeta \in \mathbb{N} \cup \{-\infty\}$. Moreover let $\zeta \cdot (-\infty) = -\infty$ for $\zeta \in \mathbb{N}$.

Let X be a variable whose value $\alpha(X)$ is defined in step 2. Then by the induction hypothesis we obtain $\alpha(Y) = \beta(Y)$ and $\alpha(Z) = \beta(Z)$. Since β is satisfying, $\beta(X) = \beta(Y) + \beta(Z)$ holds, and thus, after the execution of step 2 it holds $\alpha(X) = \beta(X)$.

Let X be a variable whose value $\alpha(X)$ is defined in step 3. Then it holds $\alpha(Z_2) \neq \emptyset$, and by the induction hypothesis we obtain $\alpha(Z_2) = \beta(Z_2)$. As for the addition $A + B = C$ with $A, B, C \in M$ and $C \neq \emptyset$ it holds $\max(C) \geq \max(A \cup B)$, it follows $\max(\beta(X)) \leq \max(\beta(Z_2))$. Hence, there is a computation path with $\alpha(X) = \beta(X)$ after the execution of step 3.

When reaching step 5 all atoms $X + Y = Z$ with defined $\alpha(X)$, $\alpha(Y)$, and $\alpha(Z)$ are satisfied by α. Thus, let $X + Y = Z$ be an atom such that one of the values $\alpha(X)$, $\alpha(Y)$, and $\alpha(Z)$ is not defined yet.
If $\alpha(Z)$ was defined and $\alpha(Z) \neq \emptyset$, then $\alpha(X)$ and $\alpha(Y)$ would be defined due to step 3 as well. If $\alpha(Z) = \emptyset$, however, then the atom $X + Y = Z$ is satisfied by setting $\alpha(X) = \emptyset$ and $\alpha(Y) = \emptyset$ respectively. Let $\alpha(Z)$ be undefined now. Then without loss of generality $\alpha(X)$ is undefined as well (due to step 2). By setting $\alpha(X) = \alpha(Z) = \emptyset$ we obtain $\alpha(X) + \alpha(Y) = \alpha(Z)$. Hence, after executing step 5 every equation of φ is satisfied. Consequently, α is a satisfying assignment.

For each variable X the value $\alpha(X)$ is defined exactly once. Let the variables be denoted as $X_1, \ldots, X_{|\mathrm{Var}_\varphi|}$ such that for $i < j$ the value $\alpha(X_i)$ is defined before $\alpha(X_j)$ is defined. Then we obtain directly from the algorithm $\max(\alpha(X_{i+1})) \leq 2 \cdot \max\left(\bigcup_{j \leq i} \alpha(X_j)\right)$. By means of a simple induction $\forall i : \max(\alpha(X_i)) \leq x2^i$ can be shown. Because of $|\mathrm{Var}_\varphi| \leq$ n the proof is complete. □

Through this result we can approach as follows: guess all assignments of variables whose ranges are subsets of $\mathcal{P}(\{0, 1, \ldots, x \cdot 2^n\})$. Test whether the respective assignment is satisfying, and return the corresponding return value. This shows the decidability of the mentioned problems. The following theorem gives information concerning the complexity.

Theorem 4.3
It holds that

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

Proof. According to lemma 2.4 it suffices to prove the two statements for $\mathrm{CSP}'(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+\})$ and $\mathrm{CSP}'([\mathbb{N}], \{+\})$ respectively.
1.: According to Lemma 4.2 the following algorithm decides $\mathrm{CSP}'(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+\})$.

- Input: φ
- Determine $n = |\varphi|$ and the greatest number x occurring in a constant.
- Guess $\alpha(X) \subseteq \{k \in \mathbb{N} \mid k \leq x \cdot 2^n\}$ for all $X \in \mathrm{Var}_\varphi$ non-deterministically.
- If α is satisfying, return 1. Otherwise return 0.

The sets $\alpha(X)$ contain at most $x \cdot 2^n$ numbers of linear length. Therefore, guessing such sets as well as checking whether a given assignment of variables is satisfying, is possible in polynomial time.

2. Almost the same algorithm can be used:

- Input: φ
- Determine $n = |\varphi|$ and the greatest number x which occurs in a constant.
- Guess $\alpha(X) = [k_1, k_2]$ with $k_1, k_2 \leq x \cdot 2^n$ for all $X \in \mathrm{Var}_\varphi$.
- Test whether α is satisfying. Return 0 or 1 correspondingly.

Here only $2 \cdot |\mathrm{Var}_\varphi|$ numbers $\leq x \cdot 2^n$ have to be guessed. This is possible in non-deterministic polynomial time. Due to the length restriction for the interval endpoints it can be tested in polynomial time whether a given atom is satisfied. Thus, the algorithm works in non-deterministic polynomial time. □

Remark 4.4
The second part of theorem 4.3 can also be proven in a shorter and easier way. Instead of showing the existence of an upper bound S such that it is sufficient to choose the interval endpoints from the set $[0, S]$, one could also decide $\mathrm{CSP}([\mathbb{N}], \{+\})$ *using the* NP*-problem* ILP *as an oracle.*
Such a proof for an even stronger result, namely $\mathrm{CSP}([\mathbb{N}], \{+, \cap\}) \in \mathrm{NP}$*, can be found in the proof of theorem 4.22.*
Yet the result $\mathrm{CSP}([\mathbb{N}], \{+\}) \in \mathrm{NP}$ *can be easily derived from lemma 4.2, which is necessary for the first part of the proof of theorem 4.3 anyway. Thus, it stood to reason to prove the statement* $\mathrm{CSP}([\mathbb{N}], \{+, \cap\}) \in \mathrm{NP}$ *also in the depicted manner.*

Now there have to be found lower bounds for both problems. We are able to show the $\leq_{\mathrm{m}}^{\log}$-hardness for NP for both problems by a reduction from an SOS-variant.
This reduction constructs sentences such that all occurring variables are mapped onto singletons by any satisfying assignment. Thus, in the case of $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+\})$ it is sufficient to use strongly restricted sentences. Hence, it suggests itself that the lower bound might not be sharp.

For $\mathrm{CSP}([\mathbb{N}], \{+\})$ however, the mentioned result yields the NP-completeness.

This is evidence that here the CSPs over finite subsets of $\mathbb{N}$ are more difficult than the corresponding CSPs over finite intervals. In section 3.2 we had observed the opposite situation.

We now define the SOS-variant mentioned before.

Definition 4.5
Let $\mathrm{MSOS} = \{x_1, \ldots, x_n, b \mid \exists a_1, \ldots, a_n \in \mathbb{N} : \sum_{i \in I} a_i x_i = b\}$.

Lemma 4.6 ([BGSW05])
MSOS *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* NP*.*

This enables us to prove the following theorem.

Theorem 4.7
$\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+\})$ *and* $\mathrm{CSP}([\mathbb{N}], \{+\})$ *are* $\leq_{\mathrm{m}}^{\log}$*-hard for* NP*.*

Proof. Let $M \in \{\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), [\mathbb{N}]\}$. We show $\mathrm{MSOS} \leq_{\mathrm{m}}^{\log} \mathrm{CSP}(\mathrm{M}, \{+\})$.

Let $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{N}$ and $b \in \mathbb{N}$.
We construct a sentence φ which is true if and only if $x_1, \ldots, x_n, b \in \mathrm{MSOS}$.

For that purpose we assume that x_i and b are given in binary representation in the MSOS-instance[9]. When constructing the sentence φ, we will use the binary representation for all occurring numbers as well.
Our construction's concept is the following: guess the values a_i with an existential quantifier and compute $a_i x_i$ by use of the shift-and-add-technique.
Firstly we describe a sentence which enforces that a variable is mapped onto $\{a_i x_i\}$ by a satisfying assignment of terms where a_i is guessed. Let thereto $b_{i,m_i} b_{i,m_i-1} \ldots b_{i,0}$ be the binary representation of x_i.
Then

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

[9]Other common encodings like the dyadic encoding can be converted into binary representation in logarithmic space.

Consider the following sentence:

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

Let α be a satisfying assignment of terms which maps each variable onto a singleton set. According to $(*)$ it holds $\alpha(R_{i,0}) = \{a_i x_i\}$ if a_i is the unique element of $\alpha(A_i)$.

Let φ be the sentence, which arises out of $\bigwedge_{i=1}^{n} \psi_i \wedge \left(\sum_{i=1}^{n} R_{i,0} = \{b\}\right)$, when all existential quantifiers are moved to the front.

Every assignment of terms α which satisfies the sentence φ maps each variable onto a singleton: otherwise α would map at least one $R_{i,0}$ onto a set containing zero or at least two elements. Then the atom $\sum_{i=1}^{n} R_{i,0} = \{b\}$ would not be satisfied. Consequently, for $M \in \{[\mathbb{N}], \mathcal{P}_{\text{fin}}(\mathbb{N})\}$ it holds that $\varphi \in \mathrm{CSP}\big(\mathrm{M}, \{=, +\}\big) \Leftrightarrow \langle \mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}, \mathrm{b} \rangle \in \mathrm{MSOS}$.

The observation that the sketched function can be computed in logarithmic space completes the proof. $\square$

4.2 Multiplication

Our results for $\mathrm{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{\times\})$ are the same as for $\mathrm{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+\})$. Nevertheless, the proofs are more complicated, which is mainly because the multiplication of non-empty sets is not monotone due to the set $\{0\}$, for which there is no additive counterpart.

For the problem $\mathrm{CSP}([\mathbb{N}], \{\times\})$ the situation is completely different from the situation we have for $\mathrm{CSP}([\mathbb{N}], \{+\})$ since the set of finite intervals is not closed under multiplication. Consequently, we are only able to show $\mathrm{CSP}([\mathbb{N}], \{\times\}) \in \Sigma_3^{\mathrm{P}}$ as an upper bound.

4.2.1 Upper Bounds

We start with the proof of the upper bounds.

CSP([ℕ], {×})

First we consider the multiplication of intervals. We show some properties, which significantly simplify deciding $\mathrm{CSP}([\mathbb{N}], \{\times\})$.

Lemma 4.8
Let $A_1, \ldots, A_m, B_1, \ldots, B_n$ be finite intervals with $\prod_{i=1}^{m} A_i = \prod_{i=1}^{n} B_i \neq \emptyset$. Then it holds that

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

If $\prod_{i=1}^{m} A_i \neq \{0\}$, then also

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

holds.

Proof. We define $\{\alpha\} =_{\text{def}} \prod_{1 \leq i \leq m, |A_i| = 1} A_i$ and $\{\beta\} =_{\text{def}} \prod_{1 \leq i \leq n, |B_i| = 1} B_i$. None of the intervals A_i, B_i is empty. Therefore, $\{\alpha\} \times \prod_{1 \leq i \leq m, |A_i| \geq 2} A_i = \{\beta\} \times \prod_{1 \leq i \leq n, |B_i| \geq 2} B_i$.
Assume $\alpha \neq \beta$. We furthermore assume without loss of generality $\alpha > \beta$. Then for all $x \in \prod_{i=1}^{m} A_i$ the number α is a divisor of x, and thus, $\alpha \mid y$ for $y \in \prod_{i=1}^{n} B_i$. Due to $\alpha > \beta$ there is a prime power p^e such that $p^e \mid \alpha$ and $p^e \nmid \beta$.
In every interval B_i with $|B_i| \geq 2$ there is a number b_i relatively prime to p: If the greatest common divisor of $\min(B_i)$ and p is greater than 1, then $\min(B_i) = rp$ for $r \in \mathbb{N}$. Thus $\gcd(r \cdot p + 1, p) = 1$.

Furthermore, $b = \beta \cdot \prod_{1 \le i \le n, |B_i| \ge 2} b_i \in \prod_{i=1}^n B_i$ and $p^e \nmid b$ hold. Consequently $\alpha \nmid b$, a contradiction.

The second statement is implied by the first. $\square$

Lemma 4.9
Let $A_1, \ldots, A_m, B_1, \ldots, B_n$ be intervals with at least two elements each. Furthermore, let

$$\max(A_1) \le \max(A_2) \le \cdots \le \max(A_m), \;\; \max(B_1) \le \max(B_2) \le \cdots \le \max(B_n),$$

and $\prod_{i=1}^m A_i = \prod_{i=1}^n B_i$. Then $\max(A_m) = \max(B_n)$.

Proof. Let $L =_{\text{def}} \prod_{i=1}^m A_i$ and $R =_{\text{def}} \prod_{i=1}^n B_i$. In addition, let the greatest elements of A_i and B_i respectively be denoted by a_i and b_i respectively.
Because of $L = R$ the second greatest elements of L and R are equal. Thus

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

We show $\max(L - \{\max(L)\}) = \left(\prod_{i=1}^{m-1} a_i\right) \cdot (a_m - 1) = \max(L) - \prod_{i=1}^{m-1} a_i$: the right equation is obvious. Furthermore, it apparently holds that $\max(L) - \prod_{i=1}^{m-1} a_i \in L - \{\max(L)\}$.
Let $x \in L - \{\max(L)\}$. Let $x_i \in A_i$ for $i = 1, \ldots, m$ such that $x = \prod_{i=1}^m x_i$. Due to $x \ne \max(L)$ there is a j such that $x_j < a_j$. Then

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

Analogously it can be seen that $\max(R - \{\max(R)\}) = \max(R) - \prod_{i=1}^{n-1} b_i$.
From $(*)$ and $\max(L) = \max(R)$ we obtain $\prod_{i=1}^{m-1} a_i = \prod_{i=1}^{n-1} b_i$. Then it follows that

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

$\square$

When a product $M = C_1 \times \cdots \times C_k$ of intervals is given and finite intervals shall be assigned to the variables $X_1, \ldots, X_n$ for $n \in \mathbb{N}$ such that $\prod_{i=1}^n X_i = M$, then for each X_i and all x from the set assigned to X_i the condition $x \le \max(\bigcup_{i=1}^k C_i)$ must hold.

By use of that property we can prove that, if $\varphi \in \text{CSP}([\mathbb{N}], \{\times\})$, then there is a non-deterministc polynomial time algorithm which finds a satisfying assignment for the sentence φ on at least one computation path. The following theorem specifies this statement.

Theorem 4.10
There is a non-deterministic polynomial time algorithm $\mathcal{A}$ with the following properties:

- *The input of $\mathcal{A}$ is a $\{\times\}$-sentence φ whose constants are exclusively finite intervals.*
- *On each computation path $\mathcal{A}$ either stops without a return value or returns a pair (ψ, α) with the following properties:*
  - *ψ is a $\{\times\}$-sentence whose constants are all finite intervals*
  - *α is an assignment of variables for ψ*
  - *if $\psi \in \text{CSP}([\mathbb{N}], \{\times\})$, then $\varphi \in \text{CSP}([\mathbb{N}], \{\times\})$*
- *If $\varphi \in \text{CSP}([\mathbb{N}], \{\times\})$, then on at least one computation path $\mathcal{A}$ returns a pair (ψ, α) such that α is a satisfying assignment of variables for ψ.*
- *Let x be the greatest number occurring in a constant of φ. For every pair (ψ, α) returned on a computation path of the algorithm and for every variable X in ψ it holds that $\max(\alpha(X)) \le x$.*

Proof. We describe the algorithm $\mathcal{A}$, which constructs an assignment α. Assume that at the beginning $\alpha(C) = C$ holds for each constant C:

1. Guess a set $K' \subseteq \mathrm{Var}_\varphi$ non-deterministically, and set $\alpha(X) = \emptyset$ for all elements $X \in K'$. Set $K = K' \cup \{\emptyset\}$.
2. For each atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$ execute the following steps.
   (a) If $K \cap \{A_1, \ldots, A_m\} \neq \emptyset \wedge K \cap \{B_1, \ldots, B_n\} \neq \emptyset$, then delete the atom a in φ.
   (b) If it holds that $K \cap \{A_1, \ldots, A_m\} \neq \emptyset \wedge K \cap \{B_1, \ldots, B_n\} = \emptyset$, or if it holds that $K \cap \{A_1, \ldots, A_m\} = \emptyset \wedge K \cap \{B_1, \ldots, B_n\} \neq \emptyset$, then terminate without a return value.
3. Guess a set $L' \subseteq \mathrm{Var}_\varphi$[10] non-deterministically and set $\alpha(X) = \{0\}$ for all $X \in L'$. Then define $L = L' \cup \{\{0\}\}$.
4. For each atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$ execute the following steps.
   (a) If $L \cap \{A_1, \ldots, A_m\} \neq \emptyset \wedge L \cap \{B_1, \ldots, B_n\} \neq \emptyset$, delete the atom a in φ.
   (b) If it holds that $L \cap \{A_1, \ldots, A_m\} \neq \emptyset \wedge L \cap \{B_1, \ldots, B_n\} = \emptyset$, or if it holds that $L \cap \{A_1, \ldots, A_m\} = \emptyset \wedge L \cap \{B_1, \ldots, B_n\} \neq \emptyset$, then terminate without a return value.
5. Guess a set $M' \subseteq \mathrm{Var}_\varphi$[11] non-deterministically.
   Set $M = M' \cup \{C \in \mathrm{Const}_\varphi \mid |\mathrm{C}| = 1\}$.
   Replace each atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$ with
   $$\Big(\prod_{1 \leq i \leq m, A_i \notin M} A_i = \prod_{1 \leq i \leq n, B_i \notin M} B_i\Big) \wedge \Big(\prod_{1 \leq i \leq m, A_i \in M} A_i = \prod_{1 \leq i \leq n, B_i \in M} B_i\Big).$$
   For each constant $\{c\} \in M$ guess the prime decomposition (if it has been guessed wrongly, terminate without a return value). Let $p_1, \ldots, p_k$ be the primes which are divisors of c. Store only the vector $(e_1, \ldots, e_k)$ for which $c = \prod_{i=1}^k p_i^{e_i}$. We also store the values $\alpha(X)$ for variables $X \in M$ that way.
6. Execute the following steps.
   (a) For each atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$ for which all occurring variables are in M apply the following rules.
      i. If $\alpha(A_i)$ is defined for every i, then there is a vector $(e_1, \ldots, e_k)$ such that $\prod_{i=1}^k p_i^{e_i}$ is the unique number $\prod_{i=1}^m \alpha(A_i)$. The vector is computed as the componentwise sum of the vectors for the $\alpha(A_i)$. For each B_i with undefined $\alpha(B_i)$ do the following: Guess a vector $(e'_1, \ldots, e'_k)$ with $e'_j \leq e_j$ non-deterministically. Store $\alpha(B_i)$ as $(e'_1, \ldots, e'_k)$.
      ii. Proceed analogously if $\alpha(B_i)$ is defined for every i.
   (b) For each atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$ for which all occurring variables are not in M apply the following rules.
      i. If all $\alpha(A_i) = [a_i, a'_i]$ are defined, then for each B_i with undefined $\alpha(B_i)$ guess two numbers $s_1 < s_2$ with $s_2 \leq \max(\{a'_1, \ldots, a'_m\})$, and define $\alpha(B_i) =_{\mathrm{def}} [s_1, s_2]$.
      ii. Proceed analogously if all $\alpha(B_i)$ are defined.
   (c) If the value $\alpha(X)$ for a variable X has been defined in the steps 6a and 6b, execute 6 again.
7. For each undefined variable X set $\alpha(X) =_{\mathrm{def}} \emptyset$.[12]

[10] Note that every atom with a variable in K has been deleted. Thus, Var_φ does not contain any elements in K.
[11] The variables in M will be mapped onto singletons by α.
[12] This step is unnecessary since there is a computation path on which the corresponding variables have already been defined to be empty in step 1 and removed afterwards. Despite of that this step simplifies the argumentation at some later point.

8. For every atom $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$ where all occurring variables are in M, test whether it is satisfied. Here it suffices to compute the sum of the respective vectors of exponents.

   If a is not satisfied, terminate without a return value. Otherwise remove a. Let the sentence constructed that way be denoted by ψ.

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

The proof is complete as soon as the following four statements have been proven:

1. For each sentence φ returned on some computation path, $\varphi \notin \mathrm{CSP}([\mathbb{N}], \{\times\})$ implies $\psi \notin \mathrm{CSP}([\mathbb{N}], \{\times\})$.
2. If $\varphi \in \mathrm{CSP}([\mathbb{N}], \{\times\})$, then on at least one computation path $\mathcal{A}$ returns a pair (ψ, α) such that α is a satisfying assignment of variables for ψ.
3. $\mathcal{A}$ works in non-deterministic polynomial time.
4. For each pair (ψ, α) returned on a computation path of $\mathcal{A}$ and each variable X in ψ, it holds $\max(\alpha(X)) \leq x$.

1.) If the input sentence is not in $\mathrm{CSP}([\mathbb{N}], \{\times\})$, then obviously the sentence computed after the execution of step 4 is not in $\mathrm{CSP}([\mathbb{N}], \{\times\})$ either.

For intervals $A_1, \ldots, A_m, B_1, \ldots, B_n$ it holds:

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

Hence, if the sentence has a satisfying assignment after step 5, then also before that step.

In step 8 it is tested whether all atoms whose variables and constants are all in M are satisfied by the assignment constructed so far. All these variables and constants do not occur in any other atom. Hence, if φ is not true, then either the conjunction of the already mentioned atoms is not true, or this conjunction is true, but the rest of the sentence is not. In the first case no sentence is returned whereas in the second case the returned sentence is not true. This proves the statment.

2.) Let $\varphi \in \mathrm{CSP}([\mathbb{N}], \{\times\})$ with a satisfying assignment of variables β. Then there is a computation path on which $\mathcal{A}$ guesses the sets K, L, M such that for all variables X of the input sentence

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

$\mathcal{A}$ does not terminate on this computation path until step 5 inclusively. We now consider the sentence which $\mathcal{A}$ has computed after the execution of step 4. This sentence is satisfied by β.

For all remaining variables the set $\beta(X)$ is not empty and does not equal $\{0\}$. Due to that and lemma 4.8 the assignment β also satisfies the sentence generated by $\mathcal{A}$ after the execution of step 5.

There is obviously a computation path on which for each variable X for which $\alpha(X)$ has been defined in step 6a $\alpha(X) = \beta(X)$.
Because of lemma 4.9 this also holds for any variable Y for which $\alpha(Y)$ has been defined in step 6b.

Hence, when reaching step 7 each atom in which for all variables X the value $\alpha(X)$ is defined is satisfied by α.

For all other atoms $a = (A_1 \times \cdots \times A_m = B_1 \times \cdots \times B_n)$ there is at least one variable A_i and at least one variable B_j such that $\alpha(A_i)$ and $\alpha(B_j)$ are not defined yet. Otherwise all variables in a would have a defined function value under α due to step 6a and step 6b.
By setting $\alpha(A_i) = \emptyset$ and $\alpha(B_j) = \emptyset$ the assignment α satisfies the atom a. Consequently, α satisfies the complete sentence and in particular also the returned sentence ψ.

3.) The steps 6a and 6b are obviously executed $|\mathrm{Var}_\varphi|$ times at most, where φ denotes the input sentence. Hence, the only steps of the algorithm for which it is not obvious that they can be executed in non-deterministic polynomial time are the steps 6a and 8.
Let e be the greatest exponent which occurs in the prime decomposition of the number c for a constant $C = \{c\}$. Let $X_1, \dots, X_r$ be the variables of M such that for $i < j$ the value $\alpha(X_i)$ is defined before $\alpha(X_j)$ is defined. Let farther x_i denote the unique element in $\alpha(X_i)$.

It can be shown inductively that the greatest exponent in the prime decomposition of x_i is not greater than $e \cdot 2^i$.
Particularly the sets $\alpha(X)$ can be computed in polynomial time and stored in polynomial space. Now it is obvious that also step 8 can be executed in polynomial time.

4.) All the atoms with variables in M have been deleted. Hence, the statement holds due to step 6b. $\square$

Thus, in order to decide whether $\varphi \in \mathrm{CSP}([\mathbb{N}], \{\times\})$, it is sufficient to test for a pair (ψ, α) returned by $\mathcal{A}$ whether α is a satisfying assignment for ψ.
Therefore, we define the following problem.

Definition 4.11
Let IntervalEquality be the set

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

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

Proof. Let $A_1, \dots, A_m, B_1, \dots, B_n$ be non-empty intervals with $|A_i|, |B_j| \geq 2$.
It holds

$$\begin{aligned}\prod_{i=1}^{m} A_i = \prod_{i=1}^{n} B_i \Leftrightarrow & \forall x_1 \in A_1 \dots \forall x_m \in A_m \forall y_1 \in B_1 \dots \forall y_n \in B_n \\ & \exists x_1' \in B_1 \dots \exists x_n' \in B_n \exists y_1' \in A_1 \dots \exists y_m' \in A_m \\ & \prod_{i=1}^{m} x_i = \prod_{i=1}^{n} x_i' \wedge \prod_{i=1}^{n} y_i = \prod_{i=1}^{m} y_i'.\end{aligned}$$

As the x_i, y_i, x_i', y_i' all have polynomial length, $A \in \forall^{\mathrm{p}}\exists^{\mathrm{p}}\mathrm{P} = \Pi_2^{\mathrm{p}}$. $\square$

With this the following upper bound for $\mathrm{CSP}([\mathbb{N}], \{\times\})$ can be proven.

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

Proof. According to theorem 4.10 and lemma 4.12 the problem $\mathrm{CSP}([\mathbb{N}], \{\times\})$ can be decided by an NP-algorithm with Π_2^{p}-oracle. $\square$

Remark 4.14
Our decision algorithm for IntervalEquality *tests whether two products of intervals are equal by considering all elements of the two products. Maybe this can be done more efficiently. In lemma 4.9 we have shown that for an equation of products of intervals the maximal upper interval endpoint is the same in both products. Perhaps it can be shown a more general result such that an algorithm only has to consider the interval endpoints of the intervals in order to decide* IntervalEquality.

CSP($\mathcal{P}_{\text{fin}}(\mathbb{N}), \{\times\}$)

We use an abbreviating notation for the set of prime divisors of numbers occurring in a sentence:

Definition 4.15
Let φ be a $\{\times\}$-sentence. We define

$$P_\varphi = \{p \mid \ p \text{ prime and } p \mid c \text{ for a } 0 \neq c \in C,\ C \in \text{Const}_\varphi\}.$$

We obtain directly from the definition:

Lemma 4.16
Let φ be a $\{\times\}$-sentence and $n =_{def} |\varphi|$. Then $|P_\varphi| \in O(n^2)$.[13]

The following lemma significantly shrinks the space in which we have to search a satisfying assignment.

Lemma 4.17
Let $\varphi \in \text{CSP}'(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{\times\})$ and let α be a satisfying assignment of variables for φ. Define β as follows: Set $\beta(X) = \emptyset$ if $\alpha(X) = \emptyset$, and set

$$\begin{aligned}\beta(X) &= \Big\{y \mid \exists x \in \alpha(X) :\ y = \frac{x}{\prod_{p \in \mathbb{P} - P_\varphi,\ p^e \mid x,\ p^{e+1} \nmid x} p^e}\Big\} \\ &= \{y \mid \exists x \in \alpha(X) :\ y \textit{ can be received from } x \textit{ by cutting all prime factors } p \notin P_\varphi\},\end{aligned}$$

if $\alpha(X) \neq \emptyset$. Then β is a satisfying assignment of variables.

Proof. Let $A \times B = C$ be an arbitrary atom in φ. Then $\alpha(A) \times \alpha(B) = \alpha(C)$. If $\emptyset \in \{\alpha(A), \alpha(B), \alpha(C)\}$, then obviously also β satisfies the atom. Otherwise

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

□

We can further reduce the size of the space in which we have to search a satisfying assignment.

Lemma 4.18
Let $\varphi \in \text{CSP}'(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{\times\})$ with $n =_{def} |\varphi|$. Define

$$\zeta = \max(\{e \mid \ p^e \mid c \textit{ for } p \in P_\varphi \textit{ and } c \in C \textit{ for some } C \in \text{Const}_\varphi\ \}).$$

Then there is a satisfying assignment of variables α for φ with

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

Proof. We present a non-deterministic algorithm which returns an assignment of variables satisfying the requirements mentioned above. Thereto the algorithm successively constructs a total mapping $\alpha : \text{Var}_\varphi \cup \text{Const}_\varphi \to \mathcal{P}_{\text{fin}}(\mathbb{N})$, from which we will eventually receive the already mentioned assignment of variables.

[13] It can be obviously found a better upper bound, namely $O(n)$. Yet the bound mentioned above is sufficent for our purposes.

1. Set $\alpha(C) = C$ for all constants C.
2. Guess a subset K' of Var_φ non-deterministically and set $\alpha(X) = \emptyset$ for all $X \in K'$. Set $K = K' \cup \{\emptyset\}$.
3. Delete all atoms $A \times B = C$ in φ for which $A, C \in K$ or $B, C \in K$.

   If for an atom $A \times B = C$ it holds that $(C \in K \wedge A, B \notin K) \vee (\{A, B\} \cap K \neq \emptyset \wedge C \notin K)$, then terminate the computation without a return value.
4. Guess a subset L' of $\mathrm{Var}_\varphi - \mathrm{K}$ non-deterministically and set $\alpha(X) = \{0\}$ for all $X \in L'$. Set $L = L' \cup \{\{0\}\}$.
5. Delete each atom $A \times B = C$ in φ for which $A, C \in L$ or $B, C \in L$.

   If there is an atom $A \times B = C$ with $(C \in L \wedge A, B \notin L) \vee (\{A, B\} \cap L \neq \emptyset \wedge C \notin L)$, then terminate the computation without a return value.
6. Execute the following steps:
   (a) For every atom $A \times B = C$ for which $\alpha(C)$ is defined[14] and $\alpha(A)$ is still undefined, guess a non-empty subset $S \neq \{0\}$ of

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

   and set $\alpha(A) = S$. Proceed analogously for B.

   (b) For each atom $A \times B = C$ with defined values $\alpha(A)$ and $\alpha(B)$ and undefined $\alpha(C)$, set $\alpha(C) = \alpha(A) \times \alpha(B)$.

   (c) If $\alpha(X)$ for a variable X has been defined in step 6a or 6b, then execute step 6 again.
7. For all $X \in \mathrm{Var}_\varphi$ with undefined $\alpha(X)$ set $\alpha(X) = \emptyset$[15].
8. Set $\alpha =_{\mathrm{def}} \alpha_{|\mathrm{Var}_\varphi}$. If α is satisfying, return α. Otherwise terminate without a return value.

We show that the algorithm returns a satisfying assignment on at least one computation path:

Let β be a satisfying assignment of variables for φ such that for every variable X the set $\beta(X)$ contains only numbers whose prime divisors are in P_φ. Such an assignment exists according to lemma 4.17. Then there is a computation path on which the algorithm above chooses the sets K and L such that for all variables X

$$X \in K \Leftrightarrow \beta(X) = \emptyset \qquad \text{and} \qquad X \in L \Leftrightarrow \beta(X) = \{0\}.$$

Let α denote the assignment which is constructed on the computation path sketched above. Until step 5 inclusively the two assignments α and β are equal for all the variables X for which $\alpha(X)$ has been defined so far. Hence, the atoms deleted during the steps 1 to 5 are satisfied by any assignment of terms ζ with $\zeta(X) = \alpha(X)$ for variables with an already defined $\alpha(X)$. Thus, these atoms require no further consideration.

In step 6 the value $\alpha(X)$ is defined only for those variables X which exclusively occur in atoms $A \times B = C$ for which $\beta(A), \beta(B), \beta(C) \notin \{\emptyset, \{0\}\}$[16]. Thus, it can be shown inductively that there is a computation path on which after step 6 the assignments α and β are equal for all variables X for which $\alpha(X)$ has been defined so far.
In particular, for every atom $A \times B = C$ for which the values $\alpha(A)$, $\alpha(B)$ and $\alpha(C)$ are defined after step 6 it holds that $\alpha(A) \times \alpha(B) = \alpha(C)$. Let $A \times B = C$ be an atom such that for at least one of the variables there has not yet been defined a value under α. Then $\alpha(C)$ and at least one of the two values $\alpha(A)$ and $\alpha(B)$ have not been defined yet: Assume that this is not so. Then

[14]Note that in the following $\alpha(C) \notin \{\emptyset, \{0\}\}$.

[15]This step is actually unneccessary since for any set of variables which are mapped onto $\emptyset$ there is a corresponding computation path in which these variables have already been mapped onto $\emptyset$ and deleted afterwards. Nevertheless, this step simplifies the argumentation at some later point.

[16]Note that some atoms of the input sentence have already been deleted

- $\alpha(C)$ has been defined wherefore according to step 6a also $\alpha(A)$ and $\alpha(B)$ have been defined,
- or $\alpha(A)$ and $\alpha(B)$ have been defined yet wherefore according to step 6b also $\alpha(C)$ has been defined.

We respectively obtain a contradiction.
Hence, α is satisfying when we set $\alpha(X) = \emptyset$ for any variable X not defined after step 6. This is done in step 7.

It remains to show $\forall X \in \mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi : \ \alpha(\mathrm{X}) \subseteq \left\{ \prod_{\mathrm{p_i} \in \mathrm{P}_\varphi} \mathrm{p_i^{e_i}} \mid \forall \mathrm{i} : \ 0 \leq \mathrm{e_i} \leq \zeta \cdot 2^{\mathrm{n}} \right\}$. Let the variable $X_1, \ldots, X_{|\mathrm{Var}_\varphi|}$ be indexed such that for $i < j$ the value $\alpha(X_i)$ is defined before $\alpha(X_j)$ is defined. Inductively it can be easily shown that for all $j = 1, \ldots, |\mathrm{Var}_\varphi|$ it holds that $\alpha(X_j) \subseteq \left\{ \prod_{p_i \in P} p_i^{e_i} \mid \forall i : \ e_i \leq \zeta \cdot 2^j \right\}$. Due to $|\mathrm{Var}_\varphi| \leq \mathrm{n}$ the proof is complete. □

The preceding lemma shows: in order to decide for a given $\{\times\}$-sentence whether it is true, it suffices to test finitely many assignments of variables with regard to whether they are satisfying. That yields the decidability of $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\times\})$.

Yet the lemma also makes a statement with respect to the size of the set of assignments which have to be tested. Thereby we can find a concrete complexity class as an upper bound for $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\times\})$.

Here the following aspect is important: It might be that for every satisfying assignment α there is a variable X such that $\alpha(X)$ contains some superexponential number. But according to the preceding lemma, for each variable X the set $\alpha(X)$ contains only exponentially many elements.

Theorem 4.19
It holds $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\times\}) \in \mathrm{NEXP}$.

Proof. According to lemma 4.18 the following algorithm decides the problem.

- Input: $\{\times\}$-sentence φ'.
- Compute $\varphi = f(\varphi')$ for the FL-function f which reduces $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\times\})$ to the problem $\mathrm{CSP}'(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\times\})$. Thus, all atoms in φ are of the form $A \times B = C$.
- Determine $n = |\varphi|$ and $\zeta = \max(\{e \mid \exists C \in \mathrm{Const}_\varphi \, \exists \mathrm{c} \in \mathrm{C} \, \exists \mathrm{p} \in \mathbb{P} : \ \mathrm{p^e} \mid \mathrm{c}\})$.
- Compute the set $P_\varphi = \{p_1, \ldots, p_{|P_\varphi|}\}$.
- For each variable X in φ guess a set $S \subseteq \left\{ \prod_{i=1}^{|P_\varphi|} p_i^{e_i} \mid \forall i : \ 0 \leq e_i \leq \zeta \cdot 2^n \right\} \cup \{0\}$ and set $\alpha(X) = S$. Set farther $\alpha(C) = C$ for all $C \in \mathrm{Const}_\varphi$.
- If there is an atom $A \times B = C$ in φ with $\alpha(A) \times \alpha(B) \neq \alpha(C)$, then return 0. Otherwise return 1.

The algorithm guesses $|\mathrm{Var}_\varphi| \in \mathrm{O(n)}$ subsets S of $\left\{ \prod_{p_i \in P_\varphi} p_i^{e_i} \mid \forall i : \ e_i \leq \zeta \cdot 2^n \right\} \cup \{0\}$ for $\zeta \in O(2^n)$. Note that the sets S contain exponentially many exponentially long numbers. Hence, the algorithm can be computed in $2^{p(n)}$ computation steps for an appropriate polynomal p. □

4.2.2 Lower Bounds

For $M \in \{[\mathbb{N}], \mathcal{P}_{\mathrm{fin}}(\mathbb{N})\}$ we prove the $\leq_{\mathrm{m}}^{\mathrm{log}}$-hardness of $\mathrm{CSP(M}, \{\times\})$ for NP. For $M = \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ this could also be shown over a reduction from $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+\})$ since the earlier proven reduction MSOS $\leq_{\mathrm{m}}^{\mathrm{log}}$ $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+\})$ yields a sentence whose constants are either singletons or empty. This can be seen at a later point after the proof of lemma 4.24. Because $[\mathbb{N}]$ is not closed under multiplication, however, we cannot proceed that way for $M = [\mathbb{N}]$.
Therefore, we show MSOS $\leq_{\mathrm{m}}^{\mathrm{log}}$ $\mathrm{CSP(M}, \{\times\})$ parallel for both cases, and thus obtain:

Theorem 4.20
For $M \in \{[\mathbb{N}], \mathcal{P}_{\text{fin}}(\mathbb{N})\}$ the problem $\text{CSP}(\text{M}, \{\times\})$ *is $\leq_{\text{m}}^{\log}$-hard for* NP.

Proof. Proceeding similarly to the proof of theorem 4.7 we show MSOS $\leq_{\text{m}}^{\log}$ $\text{CSP}(\text{M}, \{\times\})$.
Let $a_1, \ldots, a_k, b \in \mathbb{N}$. We construct a $\{\times\}$-sentence φ such that $\varphi \in \text{CSP}(\text{M}, \{\times\})$ if and only if there are $x_1, \ldots, x_k \in \mathbb{N}$ with $\sum_{i=1}^{k} x_i a_i = b$.
It holds that $\sum_{i=1}^{k} x_i a_i = b \Leftrightarrow \prod_{i=1}^{k} \left(2^{x_i} 2^{a_i}\right) = 2^b$. For a short representation of the numbers 2^s for $s \in \mathbb{N}$ in φ we use a square-and-multiply-algorithm. Thereto let $b_m \ldots b_0$ be the binary representation of s. Then $2^s = 2^{\sum_{i=0}^{m} b_i 2^i} = \prod_{0 \leq i \leq m} \left(2^{2^i}\right)^{b_i} = \prod_{0 \leq i \leq m, b_i = 1} 2^{2^i}$. Let in the following $b_{i,m_i} \ldots b_{i,0}$ be the binary representation of a_i. Furthermore, $b_{k+1,m_{k+1}} \ldots b_{k+1,0}$ denotes the binary representation of b.

For $i \in \{1, \ldots, k+1\}$ we construct a sentence ψ_i. This sentence contains a variable A_i which is mapped onto 2^{a_i} in case $i \in \{1, \ldots, k\}$ and onto 2^b in case $i = k+1$ by every satisfying assignment. For that purpose let $m = \max(m_1, \ldots, m_{k+1})$.
Set $\chi =_{\text{def}} \exists Y_m \ldots \exists Y_0 \left(Y_0 = \{2\}\right) \wedge \bigwedge_{j=0}^{m-1} \left(Y_{j+1} = Y_j \times Y_j\right)$ and $\psi_i =_{\text{def}} \exists A_i \big(A_i = \prod_{0 \leq j \leq m_i, b_j = 1} Y_j\big)$.
Let φ be the sentence which is received from $\exists X_1 \ldots \exists X_k \left(\chi \wedge \bigwedge_{i=1}^{k+1} \psi_i \wedge A_{k+1} = \prod_{i=1}^{k} \left(A_i \times X_i\right)\right)$ when all the existential quantifiers are moved to the front.

Let α be a satisfying assignment of variables for φ. Obviously α maps every variable Y_j and each variable A_i onto a singleton. If there was a variable X_i not mapped onto a singleton, then the atom $A_{k+1} = \prod_{i=1}^{k} (A_i \times X_i)$ would not be satisfied. Hence $W_\alpha \subseteq \{\mathbb{N}\}$.

Thus, it suffices to show that there are $x_1, \ldots, x_n \in \mathbb{N}$ with $\prod_{i=1}^{k} 2^{x_i} 2^{a_i} = 2^b$ if and only if there is a satisfying assignment α with $W_\alpha \subseteq \{\mathbb{N}\}$ for φ.
"$\Rightarrow$": Let $x_1, \ldots, x_n \in \mathbb{N}$ such that $\prod_{i=1}^{k} 2^{x_i} 2^{a_i} = 2^b$ holds. Then each assignment of terms with $X_i \mapsto \{2^{x_i}\}$ satisfies the sentence φ.
"$\Leftarrow$": Under every assignment α it holds that $\alpha(A_i) = \{2^{a_i}\}$ for $i = 1, \ldots, k$ and $\alpha(A_{k+1}) = \{2^b\}$. If we denote the unique element in $\alpha(X_i)$ by x_i', then $\prod_{i=1}^{k} x_i' 2^{a_i} = 2^b$. Particularly the x_i' are powers of two and for $x_1, \ldots, x_n$ with $x_i = \log_2 x_i'$ it holds $\prod_{i=1}^{k} 2^{x_i} 2^{a_i} = 2^b$.

Since the numbers occurring in the input are already given in binary representation, the sentence φ can be computed in logarithmic space. □

4.3 Addition and Intersection

In contrast to the variant over arbitrary finite subsets of $\mathbb{N}$, which will be shortly discussed in section 4.5, an upper bound for $\text{CSP}([\mathbb{N}], \{+, \cap\})$ can be found quite simple: by use of integer linear programs (ILPs) we show $\text{CSP}([\mathbb{N}], \{+, \cap\}) \in \text{NP}$.
Furthermore, we obtain the NP-hardness from the NP-hardness of $\text{CSP}([\mathbb{N}], \{+\})$.

We define a variant of integer linear programs, which is known to be in NP:

Definition 4.21
Let ILP $=_{\text{def}} \left\{(A, b) \mid A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}^m, m, n \in \mathbb{N}^+, \exists x \in \mathbb{N}^n : Ax \leq b\right\}$ where $(d_1, \ldots, d_m) \leq (e_1, \ldots, e_m)$ if and only if for all $i = 1 \ldots, m$ the condition $d_i \leq e_i$ holds.

Theorem 4.22
$\text{CSP}([\mathbb{N}], \{+, \cap\}) \in \text{NP}$.

Proof. We prove the statement by specifying a non-deterministic polynomial time algorithm. According to lemma 2.4 it suffices to show $\text{CSP}'([\mathbb{N}], \{+, \cap\}) \in \text{NP}$.

Hence, let φ be a $\{+, \cap\}$-sentence with $\text{Const}_\varphi \subseteq [\mathbb{N}]$ such that each atom is of the form $X \oplus Y = Z$ for $X, Y, Z \in \text{Var}_\varphi \cup \text{Const}_\varphi$ and $\oplus \in \{+, \cap\}$.

Guess a subset M of variables in φ non-deterministically, and replace each occurrence of a variable in M with the constant $\emptyset$. Delete each atom $X \oplus Y = Z$ for $\oplus \in \{\cap, +\}$ if $Z = \emptyset$ and $(X = \emptyset \vee Y = \emptyset)$. If no atom remains, return $\{0\} + \{0\} = \{0\}$. If for an atom $X \oplus Y = Z$ for $\oplus \in \{\cap, +\}$ the condition

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

holds, then return $\{0\} + \{0\} = \{1\}$. In all other cases return the possibly modified sentence.

Thus, $\varphi \in \mathrm{CSP}'([\mathbb{N}], \{+, \cap\})$ if and only if on at least one computation path a sentence φ' has been returned and satisfies the following conditions:

- there is a satisfying assignment of variables with range $\subseteq [\mathbb{N}] - \{\emptyset\}$ for φ', and
- if there is an atom $X \oplus Y = Z$ in φ' containing $\emptyset$ as a constant, then $\oplus = \cap$, $Z = \emptyset$, and $X, Y \neq \emptyset$.

The problem of testing these conditions can be solved with the help of an NP-algorithm for ILP. For each $R \in (\mathrm{Var}_\varphi \cup \mathrm{Const}_\varphi) - \{\emptyset\}$ we introduce two ILP-variables r_0, r_1. If $R = [l, u]$, set $r_0 = l$ and $r_1 = u$.

1. For each atom $X + Y = Z$ we set up the equations $x_0 + y_0 = z_0$ and $x_1 + y_1 = z_1$.
2. For each atom $X \cap Y = Z$ with $Z \neq \emptyset$ use four further variables d, e, d', e'. We express $z_0 = \max(x_0, y_0)$ and $z_1 = \min(x_1, y_1)$:
   - On $z_0 = \max(x_0, y_0)$: Add $x_0 \leq z_0$, $y_0 \leq z_0$, $z_0 = dx_0 + ey_0$, and $d + e = 1$.
   - On $z_1 = \min(x_1, y_1)$: Add $x_1 \geq z_1$, $y_1 \geq z_1$, $z_1 = d'x_1 + e'y_1$, and $d' + e' = 1$.
3. For each atom $X \cap Y = Z$ with $Z = \emptyset$ we want to express $y_1 < x_0 \vee x_1 < y_0$. Hence, we guess a bit b. If $b = 0$, we add the inequation $y_1 < x_0$. Otherwise we add $x_1 < y_0$.
4. Furthermore, for every pair of ILP-variables x_0, x_1 which describe the lower and upper end-point of some interval we add the inequation $x_0 \leq x_1$.

Thus, by construction it holds: On at least one computation path an ILP-instance (A, b) with $(A, b) \in \mathrm{ILP}$ is returned if and only if the sentence φ' has a satisfying assignment of variables which does not map any variable onto the empty set.
Hence, the algorithm decides $\mathrm{CSP}([\mathbb{N}], \{+, \cap\})$.

Since ILP $\in$ NP, the algorithm works in non-deterministical polynomial time. $\square$

Corollary 4.23
$\mathrm{CSP}([\mathbb{N}], \{+, \cap\})$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* NP.

Proof. $\mathrm{CSP}([\mathbb{N}], \{+, \cap\}) \in \mathrm{NP}$ follows from theorem 4.22. The hardness is obtained by theorem 4.7 and Lemma 2.2. $\square$

4.4 Lower Bounds for CSPs with Exactly One Arithmetic Operation and Set Operations

In this section we prove different lower bounds for CSPs over $\mathcal{P}_{\mathrm{fin}}(\mathbb{N})$ which permit at least one set operation beside exactly one arithmetical operation. More precisely we will show the Π_2^{p}-hardness for $\mathrm{CSP}\big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \otimes, \cup\}\big)$, where $\otimes \in \{+, \times\}$, and the PSPACE-hardness for the problem $\mathrm{CSP}\big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=, \otimes, \oplus\}\big)$, where $\otimes \in \{+, \times\}$ and $\oplus \in \{\cap, -\}$.

The two bounds follow almost immediately from literature, and it should be possible to improve them in at least some cases. We, however, focused mostly on other questions, particularly on the decidabiliy of the mentioned problems. The exact consideration of the lower bounds is left for further work.

The following lemma allows us to translate lower bounds for certain problems to other problems.

Lemma 4.24
It holds that $\mathrm{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=,+\} \cup \mathrm{O}) \leq_{\mathrm{m}}^{\log} \mathrm{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=,\times\} \cup \mathrm{O})$ *for* $O \subseteq \{\cap, \cup, -\}$ *with* $O \cap \{\cup, -\} \neq \emptyset$.

Proof. Let φ be given. Replace each $+$ with $\times$ in φ, and each constant $C = \{k_1, \dots, k_r\}$ with a variable which is mapped onto $\{2^{k_1}, \dots, 2^{k_r}\}$ by every assignment of terms. In the following we describe how this can be done.

For $i \in \{1, \dots, r\}$ let $b_l \dots b_0$ be the binary representation of k_i. Then, for each $i = 1, \dots, r$ it holds that $2^{k_i} = 2^{\sum_{j=0}^{l} b_j 2^j} = \prod_{j=0}^{l} \left(2^{2^j}\right)^{b_j} = \prod_{j \in [0,l], b_j = 1} 2^{2^j}$, where $2^{2^j} = \left(2^{2^{j-1}}\right)^2$.
Hence, $\{2^{k_i}\}$ can be described in the following way

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

Thus, for each assignment of terms γ for α_i it holds that $\gamma(X_i) = \{2^{k_i}\}$. The variable Y_j in α_i is now renamed $Y_{i,j}$[17]. Consider the sentence received from $\exists X_C \bigwedge_{1 \leq i \leq r} \alpha_i \wedge \Big(X_C = \bigcup_{1 \leq i \leq r} X_i\Big)$ by moving the existential quantifiers to the front and name it ψ_C.
For each assignment of terms γ for ψ_C one has $\gamma(X_C) = \{2^c \mid c \in C\}$.

Now for $C \in \mathrm{Const}_\varphi$ replace all occurrences of C in φ with X_C, build the conjunction of that sentence and $\bigwedge_{C \in \mathrm{Const}_\varphi} \psi_C$, move all existential quantifiers to the front, and name the sentence obtained that way φ'.

Due to $\forall x, y, z \in \mathbb{N} \quad \Big(x + y = z \Leftrightarrow 2^x \cdot 2^y = 2^z\Big)$ an assignment of variables γ for φ is satisfying if and only if the assignment of variables $X \mapsto \{2^x \mid x \in \gamma(X)\}$ is satisfying for φ'.

φ' can be computed in logarithmic space since the input numbers are given in binary representation.

As we use the union beside the multiplication for our construction, we have shown

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

According to lemma 2.5 it holds $\mathrm{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=,\times,\cup\} \cup \mathrm{O}) \leq_{\mathrm{m}}^{\log} \mathrm{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=,\times,-\})$. This completes the proof. □

Consider the following problem.

Definition 4.25
A natural number n is an integer expression. For two integer expressions α and β also $(\alpha + \beta)$ and $(\alpha \cup \beta)$ are integer expression.
We define the set $L(\alpha) \subseteq \mathbb{N}$ described by an integer expression α:
For $n \in \mathbb{N}$ let $L(n) =_{\text{def}} \{n\}$. For the integer expression $(\beta \oplus \gamma)$ with $\oplus \in \{+, \cup\}$ we define $L(\beta \oplus \gamma) =_{\text{def}} L(\beta) \oplus L(\gamma)$. Let $\mathrm{INEQ} =_{\text{def}} \{(\alpha, \beta) \mid \alpha, \beta$ are integer expressions and $L(\alpha) \neq L(\beta)\}$.

Meyer and Stockmeyer [SM73] prove the following lemma.

Theorem 4.26 ([SM73])
INEQ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* Σ_2^{p}.

We denote $\mathrm{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{=,+,\cup\}) \cap \{\varphi \mid \varphi$ is a $\{+,\cup\}$-sentence in which no variable occurs$\}$ by $\mathrm{constCSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+,\cup\})$.

Theorem 4.27
$\mathrm{constCSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+,\cup\})$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* Π_2^{p}.

[17] Thereby the sentence contains some redundant segments. Nevertheless it still can be computed in logarithmic space, which is sufficient for our purposes.

Proof. Lemma 4.26 yields that $\overline{\mathrm{INEQ}}$ is $\leq_{\mathrm{m}}^{\log}$-hard for Π_2^{p}. Let α, β be integer expressions. Thus, α and β are $\{+,\cup\}$-terms. Hence $(\alpha,\beta) \in \overline{\mathrm{INEQ}} \Leftrightarrow \alpha = \beta \in \mathrm{constCSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+,\cup\})$. □

Therefore, $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=,+,\cup\})$ is $\leq_{\mathrm{m}}^{\log}$-hard for Π_2^{p}. Yet it should be possible to show the existence of a better upper bound – such as the $\leq_{\mathrm{m}}^{\log}$-hardness for Σ_3^{p} – as we did not use any variables, but only constants for our reduction.
Due to lemma 4.24 $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=,\times,\cup\})$ is $\leq_{\mathrm{m}}^{\log}$-hard for Π_2^{p} as well. Thus we have proven the following corollary.

Corollary 4.28
$\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=,+,\cup\})$ *and* $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=,\times,\cup\})$ *are* $\leq_{\mathrm{m}}^{\log}$*-hard for* Π_2^{p}.

If beside one arithmetical operation there is also one of the two operations intersection and set difference available, we can show the corresponding problem to be PSPACE-hard. Thereto we slightly modify a proof given by McKenzie and Wagner [MW03] such that we get along without the union.

For the further argumentation we need the problem 3KNF-QBF, which is known to be $\leq_{\mathrm{m}}^{\log}$-complete for PSPACE.

Definition 4.29
Let 3KNF-QBF $=_{\mathrm{def}} \{F \mid F$ is a closed quantified boolean formula in 3-cnf with $F \equiv 1$ $\}$.

Theorem 4.30
The following statements hold:

1. $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+,\cap\})$ *is* $\leq_{\mathrm{m}}^{\mathrm{p}}$*-hard for* PSPACE.
2. $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\times,\cap\})$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* PSPACE.
3. $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+,-\})$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* PSPACE.
4. $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{\times,-\})$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* PSPACE.

Proof. We show the first two statements by use of a $\leq_{\mathrm{m}}^{\mathrm{p}}$- or $\leq_{\mathrm{m}}^{\log}$-reduction from 3KNF-QBF. The third statement can be shown analogously to the first statement whereas the last statement follows from the second.

1. Let $F = Q_1x_1 \ldots Q_mx_mH(x_1,\ldots,x_m)$ be a 3KNF-QBF-instance with $Q_1,\ldots,Q_m \in \{\exists,\forall\}$ and $H = \bigwedge_{j=1}^n H_j$ for an $n \in \mathbb{N}$ and formulas $H_1,\ldots,H_n$ with three literals each.

For $k = m, m-1, \ldots, 1, 0$ we construct a conjunction of atoms φ_k with variables $X_m, X_{m-1}, \ldots, X_k$ such that for all $\alpha_1,\ldots,\alpha_k \in \{0,1\}$ and each assignment of terms I the following condition holds:

$$Q_{k+1}x_{k+1} \ldots Q_mx_mH(\alpha_1,\ldots,\alpha_k,x_{k+1},\ldots,x_m) \Leftrightarrow \sum_{j=1}^{n} 3\cdot 6^j + \sum_{i=1}^{k} \alpha_i 6^{n+i} + \sum_{i=k+1}^{m} 6^{n+i} \in I(X_k).$$

We name this equivalence $(*)$. The φ_k are constructed such that for any two assignments of terms I, I' it holds $I(X_k) = I'(X_k)$. For that reason we will simply write X_k for the set $I(X_k)$ for an arbitrary assignment of terms I.
For $k = 0$ we obtain from $(*)$ that $Q_1x_1 \ldots Q_mx_mH(x_1,\ldots,x_m) \Leftrightarrow \sum_{j=1}^n 3\cdot 6^j + \sum_{i=1}^m 6^{n+i} \in X_0$.
Hence, we obtain for $r = \sum_{j=1}^n 3\cdot 6^j + \sum_{i=1}^m 6^{n+i}$ a polynomial time computable reduction function for 3KNF-QBF $\leq_{\mathrm{m}}^{\mathrm{p}}$ $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+,\cap\})$, namely

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

We first consider the case $k = m$. For $i = 1,\ldots,m$ define $a_i =_{\mathrm{def}} 6^{n+i} + \sum_{x_i \text{ in } H_j} 6^j$ and $b_i =_{\mathrm{def}} \sum_{\overline{x_i} \text{ in } H_j} 6^j$.

Then set $\varphi_m =_{\text{def}} \Big(X_m = \sum_{i=1}^m \{a_i, b_i\} + \sum_{j=1}^n \{0, 6^j, 2\cdot 6^j\}\Big)$. Note that each summand 6^j in the expression above occurs at most five times.
We show that then for all $\alpha_1, \ldots, \alpha_m \in \{0,1\}$ it holds

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

"$\Rightarrow$": Let $\gamma_i \in \{a_i, b_i\}$ with $\gamma_i = a_i$ for $\alpha_i = 1$ and $\gamma_i = b_i$ otherwise. As $H(\alpha_1, \ldots, \alpha_m)$ is true, for each clause H_j there is at least one satisfied literal, and thus, there is a γ_i in which the summand 6^j occurs. Hence, we obtain $\sum_{i=1}^m \gamma_i = \sum_{j=1}^n \beta_j 6^j + \sum_{i=1}^m \alpha_i 6^{n+i}$ with $\beta_j \in \{1,2,3\}$.
Then $\sum_{i=1}^m \gamma_i + \sum_{j=1}^n (3-\beta_j)\cdot 6^j = \sum_{j=1}^n 3\cdot 6^j + \sum_{i=1}^m \alpha_i \cdot 6^{n+i} \in X_m$ holds.
"$\Leftarrow$": Let

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

Then

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

for

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

As for each clause H_j the summand 6^j occurs at least once in $\sum_{i=1}^m \gamma_i$, one has due to the choice of the γ_i

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

For the step from k to $k-1$ assume $(*)$. For $Q_k = \exists$ we obtain

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

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

For $Q_k = \forall$ it can be argumented analogously:

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

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

Obviously the reduction function can be computed in polynomial time.

2. 3KNF-QBF $\leq_{\mathrm{m}}^{\log}$ CSP$(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{\times, \cap\})$ can be shown analogously: all occurrences + have to be replaced with $\times$ and the numbers 6^i for $i = 1, \ldots, n+m$ must be replaced with p_i where p_i is the i-th prime. Because of $\pi(i)$[18] $\in \Theta(i/\log i)$ one has $p_{n+m} \in O((n+m)^2)$ and thus $|p_{n+m}| \in O(\log(n+m))$. The occurring products are not computed in the reduction itself, but written as a product in the output sentence.
Except for that the argumentation is the same as in part 1.

3. For CSP$(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+, -\})$ the same proceeding as for CSP$(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+, \cap\})$ is possible. The intersection can be simulated by the set difference according to the proof of lemma 2.5. Thereby one can even show the $\leq_{\mathrm{m}}^{\log}$-hardness for PSPACE:

The constants occurring in the output sentence need not be computed in the reduction function, but they can be written into the output sentence by use of an shift-and-add-algorithm. Note that only numbers of the form 6^k for $k \leq n+m$ have to be described. As n and m are encoded as the number of clauses or variables in the input formula, the binary representation of numbers smaller or equal $n+m$ can be computed in logarithmic space. Hence, for all the numbers it is executed for, the shift-and-add-algorithm works in logarithmic space.

Sets with more than one element can be generated since we can simulate the union using the set difference (see the proof of lemma 2.5).

4. follows from 2. and lemma 2.5. □

In section 3 we had observed that under the assumption P $\neq$ NP there are CSPs for which the variant over $[\mathbb{N}]$ is more difficult than the variant over $\mathcal{P}_{\text{fin}}(\mathbb{N})$. Here we have the reverse situation.

In corollary 4.23 we had determined that CSP$([\mathbb{N}], \{+, \cap\})$ is $\leq_{\mathrm{m}}^{\log}$-complete for NP. According to theorem 4.30, however, CSP$(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+, \cap\})$ is $\leq_{\mathrm{m}}^{\mathrm{p}}$-hard for PSPACE. Hence, under the assumption NP $\neq$ PSPACE the problem CSP$(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+, \cap\})$ is more difficult than CSP$([\mathbb{N}], \{+, \cap\})$.

4.5 Discussion of Two Open Questions

Concerning the problems investigated up to now there remain several open questions. In this section we will discuss the two – in our opinion – most fundamental questions. Thereto we report about how we have tried to answer these questions, and try to give reasons what the main difficulties are.

1. For CSPs over arbitrary finite subsets with $O \in \{\{+\}, \{\times\}\}$ we were only able to show the $\leq_{\mathrm{m}}^{\log}$-hardness for NP as well as the upper bound NEXP. These two bounds are quite far apart from each other.

   Here we will only argue the case $O = \{+\}$ further on. The arguments for the case $O = \{\times\}$ are partially the same.

2. For CSP$([\mathbb{N}], \{+, \cap\})$ we were able to show the NP-completeness. However, for the problem CSP$(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+, \cap\})$ we could not even prove the decidability. Thus, we only know CSP$(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+, \cap\}) \in \Sigma_1$ from theorem 4.1.

Papers by Jeż and Okhotin: Jeż and Okhotin [JO10a] investigate similar questions. Among others they consider equations over variables, constants, and the addition. There the variables stand for arbitrary subsets of $\mathbb{N}$ whereas constants are "ultimately periodic" sets, i.e., sets $A \subseteq \mathbb{N}$ for which there are $d, p \in \mathbb{N}$ such that for all $x \geq d$ it holds that $x \in A \Leftrightarrow x + p \in A$.
Jeż and Okhotin investigate the complexity of testing whether there is a solution for such equation systems, and proved the Π_1-completeness of the problem.
For our purposes this result does not help directly since Jeż and Okhotin consider problems which are on the one hand more difficult to solve due to the infinite sets permitted as constants, but

[18] Here $\pi(i)$ denotes the number of primes smaller than or equal to i.

easier to solve in some sense on the other hand because the variables do not only stand for finite sets. Farther we could show that $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+\}) \in \mathrm{NEXP}$, which is a upper bound better than Π_1.

In further papers [JO10b, JO11, JO14] Jeż and Okhotin allow more operations besides the addition, namely intersection and union, but the results cannot be translated to our situation for the same reasons.

Generally, Jeż and Okhotin do not cover one of our areas with their investigations [Jez15]. In particular their previous papers are restricted to equation systems which have smallest/greatest/unique solutions [Jez15].

Sumsets: In the following we discuss the possibility $\mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+\}) \notin \mathrm{EXP}$.

Definition 4.31
Let $(G, +)$ be an abelian group and $S \subseteq G$ finite. Then S is a **sumset** if and only if there is an $A \subseteq G$ with $A + A = S$.

Croot and Lev [CL07] point out that it is not known whether there is a polynomial time algorithm which decides whether a given set is a sumset. More exact, they mention that the best known algorithm proceeds as follows: for each $s \in S$ and for all $2^{|S|}$ subsets S'' of $S' = \{s' - s \mid s' \in S\}$ compute $S'' + S''$, and compare it to S'.
For $G = \mathbb{Z}$ it is necessary and sufficient to consider the elements $s \in S \cap 2\mathbb{Z}$. For a sumset S namely, the numbers $\min(S)$ and $\max(S)$ are always even. Furthermore, $[1, 3]$ is no sumset, but $\{s - 1 \mid s \in [1, 3]\}$ is a sumset.
Croot and Lev indeed formulate the problem generally over subsets of abelian groups, but they mention explicitly that even if the group is cyclic there is no better algorithm known. In particular, no better algorithm for testing whether a finite set $S \subseteq \mathbb{Z}$ is a sumset is known.

Lemma 4.32
Let $A \subseteq \mathbb{Z}$ and $z \in \mathbb{Z}$. Then A is a sumset if and only if $A + \{2z\}$ is a sumset.

Proof. For $X \subseteq \mathbb{Z}$ it holds $X + X = A \Leftrightarrow (X + \{z\}) + (X + \{z\}) = A + \{2z\}$. □

Thus, instead of testing whether $S \subseteq \mathbb{Z}$ is a sumset, it can also be tested whether the set $S' = \left\{s - 2\left\lfloor \frac{\min(S\cup\{0\})}{2} \right\rfloor \mid s \in S\right\} \subseteq \mathbb{N}$ is a sumset.
Hence, it is not known whether $\mathrm{SUMSET} \in \mathrm{P}$ where $\mathrm{SUMSET} =_{\mathrm{def}} \{S \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N}) \mid S \text{ is a sumset}\}$. $\mathrm{SUMSET} \in \mathrm{NP}$ is obvious, though. Furthermore, obviously $\mathrm{SUMSET} \leq_{\mathrm{m}}^{\log} \mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+\})$ holds.
By use of $\{+\}$-terms with length n, however, even sets S with $|S| \in 2^{\Theta(n)}$ can be described.

We define

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

Then obviously $\text{SUCCINCT-SUMSET} \leq_{\mathrm{m}}^{\log} \mathrm{CSP}(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{+\})$.

For SAT one can also define a variant SUCCINCT-SAT. Note that SAT is complete for NP and SUCCINCT-SAT is complete for NEXP. By use of the translation lemma one obtains

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

Hence, if $\mathrm{EXP} \neq \mathrm{NEXP}$, then we obtain $P \neq \mathrm{NP}$ from the translation lemma. From $\mathrm{EXP} \neq \mathrm{NEXP}$ it follows $\mathrm{SAT} \notin \mathrm{P}$ and $\text{SUCCINCT-SAT} \notin \mathrm{EXP}$.
According to the current level of knowledge it is possible that we have the same situation for the two problems SUMSET and SUCCINCT-SUMSET, i.e., under the assumption $\mathrm{EXP} \neq \mathrm{NEXP}$ both $\mathrm{SUMSET} \notin \mathrm{P}$ and $\text{SUCCINCT-SUMSET} \notin \mathrm{EXP}$.

Upper bound for CSP($\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+,\cap\}$): We already know from theorem 4.30 that the problem CSP($\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+,\cap\}$) is $\leq_{\text{m}}^{\log}$-hard for PSPACE. As an upper bound we could only show the trivial property CSP($\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+,\cap\}$) $\in \Sigma_1$ up to know.
We try to find some evidence that proving a better upper bound is difficult:

In order to show CSP($\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+\}$) $\in$ NEXP we have presented an algorithm which proceeds as follows: first determine for which variables there are only finitely many possible values they can be mapped onto by a satisfying assignment of variables. Afterwards map all the remaining variables onto the empty set. By use of the intersection we can express though, that a variable X cannot be mapped onto $\emptyset$ by a satisfying assignment of variables without restricting the size of $\alpha(X)$ for a satisfying assignment of variables α. The following sentence is an example for that:

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

This sentence has no satisfying assignment over $\mathcal{P}_{\text{fin}}(\mathbb{N})$. The atom $X + \{1\} = X$ would only be satisfied by an assignment α with $\alpha(X) = \emptyset$. Such an assignment, however, would not satisfy the first atom.
An iterative algorithm like the algorithm in the proof of lemma 4.2, which tries to successively construct a satisfying assignment α, would first set $\alpha(X) \supseteq \{0\}$, then set $\alpha(X) \supseteq \{0,1\}$, and so on. In that case it would not terminate. However, if there is a break condition like in

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

for $c_i \in \mathbb{N}$, it would find a satisfying assignment.
It remains open if there is some bound for such an iterative algorithm such that it either terminates without having considered any numbers above this bound, or it does not terminate at all. If there was such a bound, then CSP($\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+,\cap\}$) would be decidable.

We consider a further example. Let $C_1, \ldots, C_k$ and $C'_1, \ldots, C'_{k'}$ be arbitrary constants. Let furthermore $\varphi(C_1, \ldots, C_k)$ and $\psi(C'_1, \ldots, C'_{k'})$ be terms over $\{+,\cap\}$. Consider the sentence

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

It formulates the question: Is there a sumset X with $\psi(C_1, \ldots, C_k) \subseteq X$ and

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

Thus, it has to be answered the question for the existence of a sumset S for which exponentially many requirements of the form $x \in S$ or $x \notin S$ are given.
Here it is unknown whether there is – in case of the existence of such a sumset – also a “small” such sumset, or whether it all such sumsets are “very large”.

4.6 Addition and Multiplication

As soon as both addition and multiplication are permitted in a CSP, we receive undecidable, but recursively enumerable problems. It is easy to see that we can formulate arbitrary diophantine equations.

More precisely, we prove such CSPs to be $\leq_{\text{m}}^{\log}$-complete for Σ_1.

Theorem 4.33
Let $M \in \{\mathcal{P}_{\text{fin}}(\mathbb{N}), [\mathbb{N}]\}$ and $O \subseteq \{\cup, \cap, -\}$. Then CSP$\big(\text{M}, \{=, +, \times\} \cup \text{O}\big)$ *is $\leq_{\text{m}}^{\log}$-complete for Σ_1.*

Proof. We first show:
A) For an arbitrary $\{+,\times\}$-sentence ψ it can be computed a $\{+,\times\}$-sentence ψ' such that

$$\psi' \in \text{CSP}\big(\text{M}, \{=, +, \times\}\big) \Leftrightarrow \text{there is a satisfying assignment of variables with range } \{\{a\} \mid a \in \mathbb{N}\} \text{ for } \psi.$$

Such a sentence ψ' can be constructed as follows: for each variable X in ψ append the following conjunction of atoms at the end of ψ:

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

Let α be an arbitrary assignment of terms for ψ'. If $\alpha(X) = \emptyset$ held for some variable X, the atom $X \times \{0\} = \{0\}$ would not be satisfied. If one had $\alpha(X) = \{x_1, \ldots, x_r\}$ for some variable X and an $r \in \mathbb{N} - \{0, 1\}$ such that $x_i < x_{i+1}$, then

$$\alpha(X + X) = \alpha(X) + \alpha(X) \supseteq \{x_1 + x_1, x_1 + x_2, \ldots, x_1 + x_r, x_r + x_r\}$$

would contain at least $r+1$ elements whereas $\alpha(\{2\} \times X) = \{2\} \times \alpha(X)$ contains r elements. Thus, α would not be satisfying.
Every satisfying assignment of terms for ψ', hence, maps each variable onto a singleton. Furthermore each satisfying assignment of variables for ψ' is satisfying for ψ as well.

Reversely, if there is a satisfying assignment of variables for ψ which maps each variable onto a singleton, then this assignment is obviously satisfying for ψ' as well.

B) According to the Matiyasevich-Robinson-Davis-Putnam theorem [Mat70, DPR61] there is an $n \in \mathbb{N}$ and a multivariate polynomial p with integer coefficients such that for each set $A \in \Sigma_1$ there is an $a \in \mathbb{N}$ such that

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

Now we can move the monomials with negative coefficients to the right side of the diophantine equation $p(a, x, y) = 0$. Consequently, there are multivariate polynomials l, r with non-negative integer coefficients such that

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

Due to A) we can formulate the right side of the equivalence above as a $\{+, \times\}$-sentence φ.

Note that l, r, and a are independent of the input x. As in particular for terms of the form x^e occurring in the polynomial the size of the exponent e is constant, the sentence φ can be computed in logarithmic space.

It holds that $\mathrm{CSP}\big(\mathrm{M}, \{=, +, \times\} \cup \mathrm{O}\big) \in \Sigma_1$ because of theorem 4.1. $\square$

4.7 Overview

The following tables yield an overview over the results obtained in this section.
The first table deals with the CSPs over $\mathcal{P}_{\mathrm{fin}}(\mathbb{N})$.

$\mathrm{CSP}\big(\mathcal{P}_{\mathrm{fin}}(\mathbb{N}), \{=\} \cup \mathrm{O}\big)$ with $O =$	hardness	member of
$\{+\}$	$\leq_{\mathrm{m}}^{\mathrm{log}}$-hard for NP, 4.7	NEXP, 4.3
$\{\times\}$	$\leq_{\mathrm{m}}^{\mathrm{log}}$-hard for NP, 4.20	NEXP, 4.19
$\{+, \cap\}$	$\leq_{\mathrm{m}}^{\mathrm{p}}$-hard for PSPACE, 4.30	Σ_1, 4.1
$\{+, \cup\}$	$\leq_{\mathrm{m}}^{\mathrm{log}}$-hard for Π_2^{p}, 4.28	Σ_1, 4.1
$\{+, -\}$	$\leq_{\mathrm{m}}^{\mathrm{log}}$-hard for PSPACE, 4.30	Σ_1, 4.1
$\{+, \times\}$	$\leq_{\mathrm{m}}^{\mathrm{log}}$-hard for Σ_1, 4.33	Σ_1, 4.1

The following tabular contains the results concerning CSPs over $[\mathbb{N}]$.

$\mathrm{CSP}\big([\mathbb{N}], \{=\} \cup \mathrm{O}\big)$ with $O =$	$\leq_{\mathrm{m}}^{\mathrm{log}}$-hard for	member of
$\{+\}$	NP, 4.7	NP, 4.3
$\{\times\}$	NP, 4.20	Σ_3^{p}, 4.13
$\{+, \cap\}$	NP, 4.23	NP, 4.22
$\{+, \times\}$	Σ_1, 4.33	Σ_1, 4.1

The lower bounds for CSPs over $[\mathbb{N}]$ are in general lower than those for the corresponding CSPs over $\mathcal{P}_{\text{fin}}(\mathbb{N})$. If $[\mathbb{N}]$ is closed under all allowed operations, then we know the corresponding CSP to be complete for one of the classes L, NP, and Σ_1. For the variant over $\mathcal{P}_{\text{fin}}(\mathbb{N})$ the problems are very difficult, even if less operations are permitted.

In contrast to the section before, here remain several open questions. The following are particular interesting:
Is $\text{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{\cup, \cap\})$ decidable? What is the exact complexity of $\text{CSP}(\mathcal{P}_{\text{fin}}(\mathbb{N}), \{+\})$? Does IntervalEquality belong to some class of the polynomial hierarchy lower than Π_2^{p}?

References

[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.