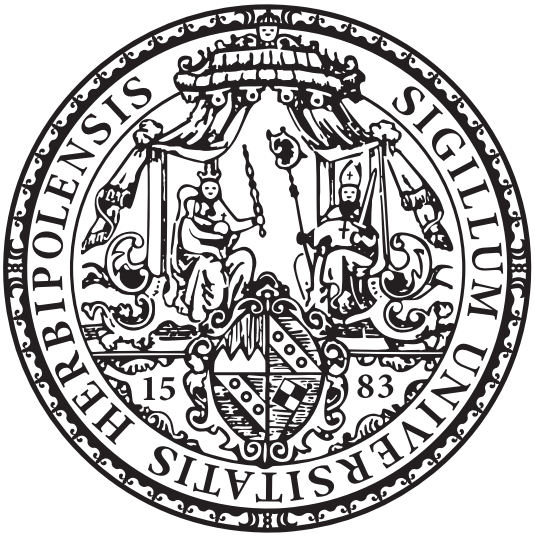
Masterarbeit

Complexity of arithmetic circuits over natural numbers with minimum and maximum

Pascal Schlereth

Abgabedatum:	15. März 2023
Betreuer:	Prof. Dr. Christian Glaßer
	Fabian Egidy, M. Sc.



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen und Komplexität

Contents

1 Introduction

In this master's thesis, it is assumed the reader is familiar with the basic concepts of graph theory and different concepts of theoretical computer science. This includes knowledge of basic complexity classes like L, P, PSPACE, EXP as well as their non-deterministic counterparts and many-one logspace respectively polynomial time reducibility.

The studies of formulas dates back to 1973 when L. J. Stockmeyer and A. R. Meyer [SM73] investigated the complexity of problems over different kinds of formulas. Formulas are expressions over a given set of inputs, e.g. numbers or formal words, connected with some operations, e.g. set union and intersection, addition or concatenation. According to these operations every formula evaluates to a solution set. Stockmeyer and Meyer studied different problems over the solution set of formulas, for example the membership problem asks for a given formula F and an instance b, if b is an element of the solution set of F. Another problem is the equivalence problem, asking for two given Formulas F_1 and F_2, if F_1 and F_2 have the same solution set.

In 2007 P. McKenzie and K. W. Wagner [MW07] considered circuits which are a generalization of formulas. Instead of an expression we consider an acyclic graph, where the vertices are either labeled as an input element or an operation. They investigated lower and upper bounds for the membership problem of circuits (and formulas) with natural numbers as inputs and operations from the set $\mathcal{O}$, where $\mathcal{O} \subseteq \{\cup, \cap, \overline{\ }, +, \times\}$. (Here $\overline{\ }$ denotes the complement of a set regarding the set of natural numbers $\mathbb{N}$.)

This thesis can be viewed as a direct expansion on this work by McKenzie and Wagner. Let $\mathcal{O} \subseteq \{\cup, \cap, \overline{\ }, +, \times\}$. The question we are trying to solve is: what are the lower and upper bounds for the complexity of the membership problem for a circuit (respectively formula) over natural numbers with operators from $\mathcal{O}$ as well as the operators min and max, whose value is defined by taking the minimum respectively maximum of the incoming set?

Chapter 2 covers the formal setup necessary for our studies. In the short chapter 3 we discuss some general strategies and techniques for tackling the membership problems proposed by this thesis. In the following three chapters, we study the different membership problems. They are split up in those problems, where only set operators are allowed (Chapter 4), where both arithmetically operators $+$ and $\times$ are allowed (Chapter 5), and lastly where only one of $+$ and $\times$ are allowed (Chapter 6). Chapter 7 summarizes the lower and upper that we have proved.

2 Definitions

This chapter covers the definitions of circuits respectively formulas as well as that of membership problems based on circuits resp. formulas. Furthermore this chapter provides the basic setup we need for the studies of the membership problems in the upcoming chapters.

2.1 Concerning circuits and formulas

The following gives a formal and proper definition of the main object studied in this thesis.

Definition 2.1 (Arithmetic circuits)**.** Let $\mathcal{O} \subseteq \{\cup, \cap, \overline{\ }, +, \times, \min, \max\}$. Then an $\mathcal{O}$-circuit $C \coloneqq (V, E, v_C, \alpha)$ is defined as:

(i) (V, E) is a finite directed acyclic graph, where each vertex's indegree is at most 2. The vertices of this graph are also called **gates**; the vertices with indegree 0 are called **input gates**. The other gates with indegree greater than 0 are called **computational gates**.

(ii) $v_C \in V$ is the designated **output gate**. In practice, this may often be "the last" gate. However, the definition allows for every gate in V to be labeled as the output.

(iii) The function $\alpha \colon V \to \mathcal{O} \cup \mathbb{N}$ assigns each gate a label, hence we call this function the **label function**. The indegree of a given gate $v \in V$ determines the possible values of $\alpha(v)$:

- If v has indegree 0, then $\alpha(v) \in \mathbb{N}$.
- If v has indegree 1, then $\alpha(v) \in \{\overline{\ }, \min, \max\}$.
- If v has indegree 2, then $\alpha(v) \in \{\cup, \cap, +, \times\}$.

(iv) The **interpretation function** $I \colon V \to \mathcal{P}(\mathbb{N})$ describes the "semantic" of the circuit by assigning each gate a set of natural numbers, representing the computed values at this point in the circuit. For $v \in V$ let u be its predecessor, if v has indegree 1, resp. u_1, u_2 its predecessors, if v has indegree 2. We define I inductively:

- If v is an input gate, then $I(v) \coloneqq \{\alpha(v)\}$.
- If $\alpha(v) = \cup$, then $I(v) \coloneqq I(u_1) \cup I(u_2)$.
- If $\alpha(v) = \cap$, then $I(v) \coloneqq I(u_1) \cap I(u_2)$.

- If $\alpha(v) = \neg$, then $I(v) \coloneqq \mathbb{N} \setminus I(u)$.
- If $\alpha(v) = +$, then $I(v) \coloneqq \{a + b \mid a \in I(u_1), b \in I(u_2)\}$.
- If $\alpha(v) = \times$, then $I(v) \coloneqq \{a \cdot b \mid a \in I(u_1), b \in I(u_2)\}$.
- If $\alpha(v) = \min$, then $I(v) \coloneqq \{a \in I(u) \mid a \leq b, \forall b \in I(u)\}$.
- If $\alpha(v) = \max$, then $I(v) \coloneqq \{a \in I(u) \mid a \geq b, \forall b \in I(u)\}$.

Conveniently we set $I(C) \coloneqq I(v_C)$. This set $I(C)$ can be viewed as *the solution* set of C. We may paraphrase this by saying "$I(C)$ is computed by C" and for the elements $b \in I(C)$ we say "b is computed by C".

This formal definition is complemented by an graphical example.

Example 2.2. Consider the following circuit C:

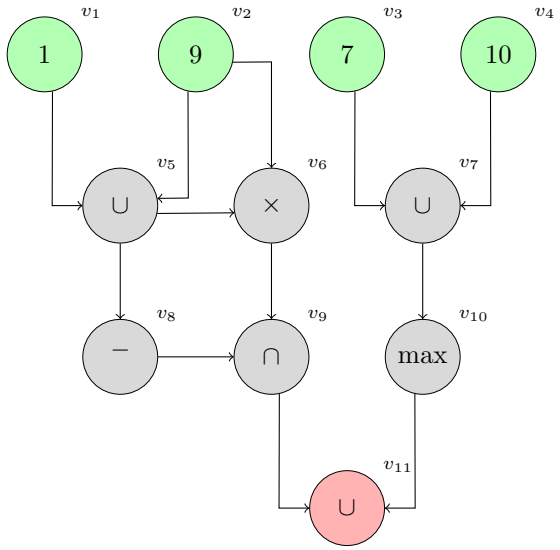


Figure 2.1: An example of a circuit.

The input gates are marked green, the computational gates are marked grey and the output gate is marked red. The values of the interpretation can be found in the following table.

$I(v_1) = \{1\}$	$I(v_5) = \{1, 9\}$	$I(v_9) = \{81\}$
$I(v_2) = \{9\}$	$I(v_6) = \{9, 81\}$	$I(v_{10}) = \{10\}$
$I(v_3) = \{7\}$	$I(v_7) = \{7, 10\}$	$I(v_{11}) = \{10, 81\}$
$I(v_4) = \{10\}$	$I(v_8) = \mathbb{N} \setminus \{1, 9\}$	$\Rightarrow I(C) = \{10, 81\}$

So the circuit in this example computes the set $\{10, 81\}$.

Based on the definition of circuits, we define the second object studied in this thesis.

Definition 2.3 (Arithmetic formulas)**.** Let $\mathcal{O} \subseteq \{\cup, \cap, \bar{}, +, \times, \min, \max\}$ and $F \coloneqq (V, E, v_F, \alpha)$ be an $\mathcal{O}$-circuit. Then F is an arithmetic formula if and only if for every gate $v \in V$ the outdegree is at most 1.

The name "formula" is indeed well chosen as every circuit satisfying the formula criterion can be written as a formula as shown in the following example:

Example 2.4. Consider the following circuit F:

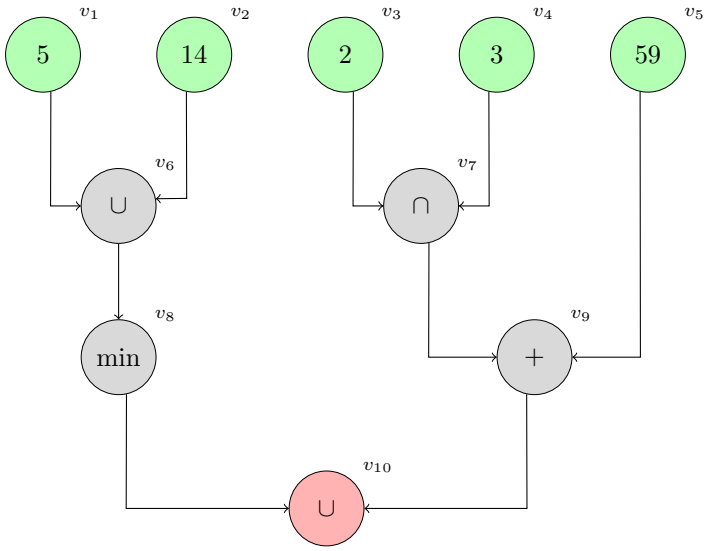


Figure 2.2: An example of a formula.

As the outdegree of every gate in this example is at most 1, F is also a formula. One can write this formula as

$$F = \min(5 \cup 14) \cup ((2 \cap 3) + 59).$$

By the way, the set computed by F is $\{5\}$. This is due to the fact that $I(v_8) = \{5\}$ and $I(v_9) = \emptyset$, as $I(v_9)$ is the sum of $\{59\}$ and an empty set.

Before continuing with the definition of the membership problem, we establish an useful notation.

Definition 2.5. Let $\mathcal{O} \subseteq \{\cup, \cap, \bar{}, +, \times, \min, \max\}$ and $C \coloneqq (V, E, v_C, \alpha)$ be a $\mathcal{O}$-circuit. For a gate $v \in V$ denote by $C_v \coloneqq (V, E, v, \alpha)$ the same circuit as C but with v being the output gate.

2.2 Membership problems

Based on circuits and formulas one can study different decision problems. For example there is the emptiness problem asking whether a given circuit/formula computes the empty set or not. Another problem, called the equivalence problem, asks, if two given circuits/formulas are computing the same set or not. However, subject of this thesis are the so called membership problems which ask for a given circuit/formula C and a given number b, if b is computed by C or not.

Definition 2.6 (Membership problem)**.** Let $\mathcal{O} \subseteq \{\cup, \cap, {}^{-}, +, \times, \min, \max\}$.

(i) Then define the membership problem of circuits regarding $\mathcal{O}$ as

$$\mathrm{MC}(\mathcal{O}) \coloneqq \{(C, b) \mid C \text{ is a } \mathcal{O}\text{-circuit, } b \in \mathbb{N} \text{ and } b \in I(C)\}\,.$$

(ii) Then define the membership problem of formulas regarding $\mathcal{O}$ as

$$\mathrm{MF}(\mathcal{O}) \coloneqq \{(F, b) \mid F \text{ is a } \mathcal{O}\text{-formula, } b \in \mathbb{N} \text{ and } b \in I(F)\}\,.$$

Instead of writing $\mathrm{MC}(\{\cup, \cap, +\})$ for example, we will forfeit the set brackets and simply write $\mathrm{MC}(\cup, \cap, +)$. The same applies for formulas.

2.3 The encoding of circuits and formulas

Let $C \coloneqq (V, E, v_C, \alpha)$ be a circuit with $V = \{v_1, v_2, \ldots, v_n\}$. As this object is basically a finite graph, we encode it using the concept of adjacency lists. For $v \in V$ denote by $l(v) \coloneqq \{w \in V \mid (v, w) \in E\}$ all successor gates of v. Then we assume C is encoded in the following way:

$$C\colon v_1, \alpha(v_1), l(v_1) - v_2, \alpha(v_2), l(v_2) - \cdots - v_n, \alpha(v_n), l(v_n) - v_C.$$

In this method of encoding there is no redundant information present.

Example 2.7. Considering once more the circuit C from Example 2.2, then the encoding of this circuit is

$$\begin{aligned} C\colon\ & (v_1, 1, v_5) - (v_2, 9, v_5, v_6) - (v_3, 7, v_7) - (v_4, 10, v_7) - (v_5, \cup, v_6, v_8) - (v_6, \times, v_9) \\ & - (v_7, \cup, v_{10}) - (v_8, {}^{-}, v_9) - (v_9, \cap, v_{11}) - (v_{10}, \max, v_{11}) - (v_{11}, \cup) - v_{11}. \end{aligned}$$

Here brackets are used for better visualization. Studying this encoding, one may notice that for a given gate v_i there are only successors v_j satisfying $j > i$. This property of a graph is called topological ordering.

Definition 2.8 (Topological ordering)**.** Let $G \coloneqq (V, E)$ with $V = \{v_1, \ldots, v_n\}$ be a finite graph.

(i) An **ordering** of the vertices V is a bijective function $\pi\colon \{1, \ldots, n\} \to V$.

(ii) If this ordering π satisfies

$$i > j \Rightarrow (\pi(i), \pi(j)) \notin E,$$

then this ordering is called **topological ordering**.

The following well-known result from algorithmic graph theory shows that for every circuit there exists an ordering of its gates, such that this ordering is a topological ordering.

Proposition 2.9
Let $G := (V, E)$ be a graph. Then there exists a topological ordering if and only if G is acyclic.

Theorem 2.10
Let $\mathcal{O} \subseteq \{\cup, \cap, \bar{}, +, \times, \min, \max\}$ and $C := (V, E, v_C, \alpha)$, such that (V, E) is a graph, $v_C \in V$ and $\alpha \colon V \to \mathbb{N} \cup \mathcal{O}$ be a label function. Then the test, if C is a circuit resp. formula and its gates are topologically ordered, is possible in logarithmic space.

Proof. We give a constructive proof by explicitly stating this algorithm. What does this algorithm have to check?

Keeping Definition 2.1 in mind, we have to check if (V, E) is acyclic. According to Proposition 2.9, if the gates of V are given in topological order, this property is automatically satisfied. We can test if (V, E) is topologically ordered by going through every gate $v \in V$ and check whether v is a successor of a gate u, where u is a gate found after v in the ordering of V. If we found such a pair of gates u, v, we have found a contradiction. If the test finds no such pair, then the gates of (V, E) are indeed topologically ordered.

Further we have to check if the function α labels the gates properly. If a gate $v \in V$ has no predecessors, then $\alpha(v)$ has to be a natural number. If v has one predecessor, then $\alpha(v)$ has to be an element of $\{\bar{}, \min, \max\}$ and if v has two predecessors, then $\alpha(v)$ has to be an element of $\{\cup, \cap, +, \times\}$. If v has more than two predecessors, then we have a contradiction. Keeping the assumption of V being topologically ordered in mind, it is sufficient to search for the predecessors of v "to the left of v" in the ordering of V.

If we want to decide the question whether C is a formula, we additionally have to check if every gate has at most 1 successor.

The resulting algorithm 2.1 for testing, if a given C is a formula, can be found on the next page. The memory space this algorithm needs is saving the value of the variable *counter* and managing two for-loops. Since the value of *counter* can not exceed 3, both of these things can be done in logarithmic space. Additionally in the case of testing for formulas, one has to test if there is more than one successor for every gate. This can be done in logarithmic space also, which concludes this proof. ■

Algorithm 2.1 Deciding, whether a given C is a formula. In case of testing C to be a circuit, delete the lines 18 and 19.

Input: Graph (V, E), $v_C \in V$ and $\alpha \colon V \to \mathbb{N} \cup \mathcal{O}$

1: **for** every $v \in V$ **do**
2: Initialize *counter* as 0. This variable counts how many
3: predecessors v has in the graph.
4: **for** every $u \in V$ with $u > v$[1] **do**
5: **if** $(u, v) \in E$ **then**
6: Return $false$, as this contradicts the topological ordering.
7: **for** every $u \in V$ with $u < v$ **do**
8: **if** $(u, v) \in E$ **then**
9: Increment *counter* by 1.
10: **if** *counter* > 2 **then**
11: Return $false$, as there can not be a gate with more than 2 predecessors.
12: **if** $\alpha(v) \in \mathbb{N}$ and *counter* $\neq 0$ **then**
13: Return $false$
14: **if** $\alpha(v) \in \{^{-}, \min, \max\}$ and *counter* $\neq 1$ **then**
15: Return $false$
16: **if** $\alpha(v) \in \{\cup, \cap, +, \times\}$ and *counter* $\neq 2$ **then**
17: Return $false$
18: **if** $|l(v)| > 1$[2] **then**
19: Return $false$
20: Return *true*, as there was no contradiction found

The consequence of this Theorem is pretty handy: every complexity class encountered in the following chapters is at least L, so we can assume a given C is always a proper circuit resp. formula, as the test is never more complex than the task itself. Additionally, we can assume the ordering of the gates to be topologically ordered. So whenever we encounter in the upcoming algorithms a *"for every $v \in V$ do"*-loop (as seen in line 1 of Algorithm 2.1 for example), we assume the gates are processed in a topologically ordered-way.

Remark. In the upcoming algorithms we will often alter a given circuit C by adding new gates to the circuit. These gates must be properly named, we have to ensure that the name of the newly established gate is not already present in C. This can be done in the following way. Read once through the names of the gates in C and store a pointer towards the lexicographically largest name. When a new gate is constructed, choose the next largest name and update the pointer. Constructing and updating this pointer consumes logarithmic space, so whenever we construct a new gate, we can name it in an efficient way.

[1] This notation means u is found after v regarding the ordering of V. Analogously $u < v$ means u is found before v.
[2] $|l(v)|$ denotes the number of successors of v.

3 Solving membership problems with minimum and maximum

Before jumping right into solving $\mathrm{MC}(\mathcal{O})$ and $\mathrm{MF}(\mathcal{O})$ for different $\mathcal{O}$, let us consider the following statement.

Theorem 3.1
Let $\mathcal{O} \subseteq \mathcal{P} \subseteq \{\cup, \cap, ^{-}, +, \times, \min, \max\}$ and $\mathcal{C}$ be a complexity class. If $\mathcal{C}$ is a lower bound for the complexity of $\mathrm{MC}(\mathcal{O})$ *resp.* $\mathrm{MF}(\mathcal{O})$*, then $\mathcal{C}$ is also a lower bound for the complexity of* $\mathrm{MC}(\mathcal{P})$ *resp.* $\mathrm{MF}(\mathcal{P})$*.*

Proof. Trivial, as every $\mathcal{O}$-circuit resp. $\mathcal{O}$-formula can be viewed as a $\mathcal{P}$-circuit resp. $\mathcal{P}$-formula. ■

In other words: Adding the possibility of having computational gates of another type can never make the solving of membership problems easier. This theorem implies a direct way to build upon the studies of McKenzie and Wagner in [MW07]. Their results are given in Table 3.1 on the next page.

Theorem 6.3 will state that $\mathrm{MC}(\cup, \cap, ^{-}, +, \min, \max) \in \mathrm{PSPACE}$, so PSPACE is an upper bound of the complexity of $\mathrm{MC}(\cup, \cap, ^{-}, +, \min, \max)$. Combining Table 3.1 and Theorem 3.1 we immediately get PSPACE as a lower bound for the complexity of $\mathrm{MC}(\cup, \cap, ^{-}, +, \min, \max)$ and the analysis of this case is complete. In fact we are going to see that most of the time the membership problem is not getting harder to solve by adding min-, max-gates. In these cases it is sufficient to cover the proofs for the upper bounds *not getting bigger* compared to the respective cases without min-, max-gates.

Theorem 6.3 shows $\mathrm{MC}(\cup, \cap, ^{-}, +, \min, \max) \in \mathrm{PSPACE}$ by stating an algorithm running in polynomial space for solving this case. The idea there is to eliminate the min-, max-gates of a given circuit C one-by-one in a topologically ordered way by explicitly computing the value of I at these gates. This is done by using the fact that there is a PSPACE-algorithm *oracle* for solving $\mathrm{MC}(\cup, \cap, ^{-}, +)$. Let v be the current min-, max-gate with predecessor u, then ask for different $i \in \mathbb{N}$ if $(C_u, i) \in \mathrm{MC}(\cup, \cap, ^{-}, +)$. Technically C_u is not a $(\cup, \cap, ^{-}, +)$-circuit, however as we eliminate the min-, max-gates in order, all possible min-, max-gates are *behind* u, so $I(u)$ is independent of any min-, max-gate.

Now more general let $\mathcal{O} \subseteq \{\cap, \cup, ^{-}, +, \times\}$ and $\mathcal{M} \coloneqq \{\min, \max\}$. We will encounter this process of explicitly eliminating min-, max-gates by computing their values several times. It leads organically to the approach of trying to show $\mathrm{MC}(\mathcal{O} \cup \mathcal{M}) \leq_m^{\mathrm{P}} \mathrm{MC}(\mathcal{O})$

resp. $\mathrm{MC}(\mathcal{O} \cup \mathcal{M}) \leq_m^{\log} \mathrm{MC}(\mathcal{O})$. If the input circuit is a formula, this process may even respect the property of formulas, resulting in a solution for $\mathrm{MF}(\mathcal{O} \cup \mathcal{M})$ instantly.

Table 3.1: Solution of the membership problem, studied in [MW07].

$\mathcal{O}$	$\mathrm{MC}(\mathcal{O})$		$\mathrm{MF}(\mathcal{O})$	
	lower bound	upper bound	lower bound	upper bound
$\cup, \cap, {}^{-}, +, \times$	NEXPTIME	?	PSPACE	?
$\cup, \cap, \quad +, \times$	NEXPTIME	NEXPTIME	NP	NP
$\cup, \quad +, \times$	PSPACE	PSPACE	NP	NP
$\cap, \quad +, \times$	P	co-R	L	L
$+, \times$	P	P	L	L
$\cup, \cap, {}^{-}, +$	PSPACE	PSPACE	PSPACE	PSPACE
$\cup, \cap, \quad +$	PSPACE	PSPACE	NP	NP
$\cup, \quad +$	NP	NP	NP	NP
$\cap, \quad +$	$\mathrm{C}_{=}\mathrm{L}$	$\mathrm{C}_{=}\mathrm{L}$	L	L
$+$	$\mathrm{C}_{=}\mathrm{L}$	$\mathrm{C}_{=}\mathrm{L}$	L	L
$\cup, \cap, {}^{-}, \quad \times$	PSPACE	PSPACE	PSPACE	PSPACE
$\cup, \cap, \quad \times$	PSPACE	PSPACE	NP	NP
$\cup, \quad \times$	NP	NP	NP	NP
$\cap, \quad \times$	$\mathrm{C}_{=}\mathrm{L}$	P	L	L
$\times$	NL	NL	L	L
$\cup, \cap, {}^{-}$	P	P	L	L
$\cup, \cap,$	P	P	L	L
$\cup,$	NL	NL	L	L
$\cap,$	NL	NL	L	L

4 Problems with only set operators

This chapter is dedicated to membership problems, where only set operators, namely $\{\cup, \cap, \overline{\ }, \min, \max\}$, are allowed as computational gates in the underlying circuit resp. formula. In section 4.1 we will show $\mathrm{MC}(\cup, \cap, \overline{\ }, \min, \max) \in \mathrm{P}$ giving us an upper bound for all these problems. After that, we consider in section 4.2 $\mathrm{MC}(\cap, \min, \max)$ and in Section 4.3 $\mathrm{MC}(\cup, \min, \max)$ as well as $\mathrm{MC}(\cup, \min)$ resp. $\mathrm{MC}(\cup, \max)$, as these problems have indeed a different complexity compared to $\mathrm{MC}(\cup, \min, \max)$. We will conclude this chapter with a quick look at $\mathrm{MC}(\min, \max)$ in section 4.4.

4.1 An upper bound

Let $C := (V, E, v_C, \alpha)$ be a $(\cup, \cap, \overline{\ }, \min, \max)$-cirucit and $b \in \mathbb{N}$. By directly computing the values of the interpretation function $I(v)$ for every gate v, we just have to ask in the end, if the given number b is in the output $I(v_C)$ or not. This may seem not possible, because applying a $\overline{\ }$-operator on a finite set yields an infinite set. However, these infinite sets can be represented by stating their complement. The upcoming Lemma shows that there exists a set W such that for every gate $v \in V$ either $I(v) \subseteq W$ or $\overline{I(v)} \subseteq W$. Moreover this Lemma gives an upper bound for the size of such an set W, guaranteeing the polynomial runtime.

Lemma 4.1
Let $C := (V, E, v_C, \alpha)$ be a $(\cap, \cup, \overline{\ }, \min, \max)$-circuit.

(i) There exists a set W satisfying $|W| \leq |V|$ and either $I(v) \subseteq W$ or $\overline{I(v)} \subseteq W$ for all $v \in V$, so the number of elements in W are polynomially bounded in $|C|$.

(ii) Let m be the biggest input element of C and $w := \max W$. Then $|w| \leq |m| + |V|$, so the size of the elements of W are polynomially bounded in $|C|$.

Proof. We will show (i) inductively.

Base Case: Define $W_0 := \bigcup_{\text{input gate } v} I(v)$ the union of all input sets. Then

$$|W_0| \leq |V| \text{ and } I(v) \subseteq W_0$$

for all input gates v, so the statement is true for all input gates v.

Inductive step: Let $v \in V$ be a computational gate with predecessor(s) u resp. u_1, u_2. There exists a set W_i satisfying $|W_i| \leq |V|$ and

$$I(u) \subseteq W_i \text{ or } \overline{I}(u) \subseteq W_i$$

resp.

$$I(u_1) \subseteq W_i \text{ or } \overline{I}(u_1) \subseteq W_i \text{ and } I(u_2) \subseteq W_i \text{ or } \overline{I}(u_2) \subseteq W_i.$$

(i) Let $\alpha(v) = \cap$ and first assume $I(u_1) \subseteq W_i$[1]. Then

$$I(v) = I(u_1) \cap I(u_2) \subseteq I(u_1) \subseteq W_i.$$

Now Assume $\overline{I(u_1)}, \overline{I(u_2)} \subseteq W_i$. Then

$$\overline{I(v)} = \overline{I(u_1) \cap I(u_2)} = \overline{I(u_1)} \cup \overline{I(u_2)} \subseteq W_i.$$

(ii) Let $\alpha(v) = \cup$ and assume $I(u_1), I(u_2) \subseteq W_i$. Then

$$I(v) = I(u_1) \cup I(u_2) \subseteq W_i.$$

Assume $\overline{I(u_1)} \subseteq W_i$[2]. Then

$$\overline{I(v)} = \overline{I(u_1) \cup I(u_2)} = \overline{I(u_1)} \cap \overline{I(u_2)} \subseteq \overline{I(u_1)} \subseteq W_i.$$

(iii) Let $\alpha(v) = \neg$-gate and assume $I(u) \subseteq W_i$. Then

$$\overline{I(v)} = I(u) \subseteq W_i.$$

Assume $\overline{I(u)} \subseteq W_i$. Then

$$I(v) = \overline{I(u)} \subseteq W_i.$$

(iv) Let $\alpha(v) = \min$ and assume $I(u) \subseteq W_i$. Then

$$I(v) = \min I(u) \subseteq W_i$$

Assume $\overline{I(u)} \subseteq W_i$. Then $I(v) = \{\min I(u)\}$ may not be a subset of W_i. However, since $I(v)$ consists of one element, we can afford to add $\min I(u)$ to the set W_i by $W_{i+1} := W_i \cup \{\min I(u)\}$. (The total number of values added through this case can not exceed the number of min-gates present in the circuit C.) Thus, through this expansion we get

$$I(v) \subseteq W_{i+1} \text{ and } |W_{i+1}| \leq |V|.$$

(v) Let $\alpha(v) = \max$ and assume $I(u) \subseteq W_i$. Then

$$I(v) = \max I(u) \subseteq W_i.$$

Assume $\overline{I(u)} \subseteq W_i$. Then.

$$I(v) = \max I(u) = \emptyset \subseteq W_i.$$

[1]The case $I(u_2) \in W_i$ is analogouos.

[2]The case $\overline{I(u_2)} \in W_i$ is analogous.

Regarding (ii), we first state $m = \max W_0$ by definition of W_0. In the above analysis we have seen, that the only case we may have to expand W_0 is $\alpha(v) = \min$ and $\overline{I(u)} \subseteq W_0$, with u being the predecessor of v. In this case $I(v) = \{n\}$, where n is the smallest number not present in $\overline{I(u)}$. This yields n is at most $m + 1$ and either m or $m + 1$ would be the maximum of the newly expanded W_1. We see the maximum of the sets W_i can increase at most by 1 compared to W_{i-1}, so the maximum of W can never be bigger than $m + |V|$, because $|V|$ is an upper bound for the number of min-gates in C. ■

Theorem 4.2
Let $\mathcal{O} \subseteq \{\cup, \cap, \overline{\ }\}$ *and* $\mathcal{M} \coloneqq \{\min, \max\}$. *Then*

$$\mathrm{MC}(\mathcal{O} \cup \mathcal{M}) \in \mathrm{P}$$

Proof. Without loss of generality assume $\mathcal{O} = \{\cup, \cap, \overline{\ }\}$. Let $C \coloneqq (V, E, v_C, \alpha)$ be a $(\mathcal{O} \cup \mathcal{M})$-circuit and $b \in \mathbb{N}$. Algorithm 4.1 decides the question, if $b \in I(C)$ as this algorithm computes the values of $I(v)$ explicitly (resp. those of $\overline{I(v)}$) and saving them in the list entry $L[v]$. In the end, the algorithm asks whether $b \in L[v]$ and gives the right answer back, depending on if $L[v]$ represents $I(v)$ or $\overline{I(v)}$.

The algorithm iterates through every gate of the circuit and performs some kind of set operations. (Except for $\alpha(v) = \overline{\ }$, where the previous set is just copied.) The size of these sets is polynomially bounded in $|C|$, as the number of elements in these sets are polynomially bounded as well as the size of these elements. (See Lemma 4.1.) This implies that computing $L[v]$ is doable in polynomial time for every gate $v \in V$. All in all this gives polynomial runtime for Algorithm 4.1, which concludes this proof.

Algorithm 4.1 Solving $\mathrm{MC}(\cup, \cap, \overline{\ }, \min, \max)$

Input: $(\cup, \cap, \overline{\ }, \min, \max)$-circuit $C \coloneqq (V, E, v_C, \alpha)$ and $b \in \mathbb{N}$.
1: Initialize an empty list L, which will consist of elements (M, t) with $M \subset \mathbb{N}$
2: and $t \in \{0, 1\}$, where 1 indicates $I(v) = M$ and 0 indicates $\overline{I(v)} = M$.
3: **for** every input gate $v \in V$ **do**
4: Set $L[v] \coloneqq (\{\alpha(v)\}, 1)$.
5: **for** every computational gate $v \in V$ **do**
6: Find the predecessor(s) u resp. u_1, u_2 of v.
7: Further denote $L[u] = (M, t)$ resp. $L[u_1] = (M_1, t_1)$ and $L[u_2] = (M_2, t_2)$.
8: **if** $\alpha(v) = \cup$ **then**
9: **if** $t_1 = t_2 = 1$ **then**
10: Set $L[v] \coloneqq (M_1 \cup M_2, 1)$.
11: **else if** $t_i = 1$ and $t_j = 0$[3] **then**
12: Set $L[v] \coloneqq (M_j \setminus M_i, 0)$.
13: **else if** $t_1 = t_2 = 0$ **then**
14: Set $L[v] \coloneqq (M_1 \cap M_2, 0)$.
15: **else if** $\alpha(v) = \cap$ **then**

[3]Here and in the following $i, j \in \{0, 1\}$ and $i \neq j$. This notation shortens the length of the algorithm.

16: **if** $t_1 = t_2 = 1$ **then**
17: Set $L[v] \coloneqq (M_1 \cap M_2, 1)$.
18: **else if** $t_i = 1$ and $t_j = 0$ **then**
19: Set $L[v] \coloneqq (M_i \setminus M_j, 1)$.
20: **else if** $t_1 = t_2 = 0$ **then**
21: Set $L[v] \coloneqq (M_1 \cup M_2, 0)$.
22: **else if** $\alpha(v) = \bar{}$ **then**
23: Set $I(v) \coloneqq (M, 1 - t)$.
24: **else if** $\alpha = \min$ **then**
25: **if** $t = 1$ **then**
26: Set $L[v] \coloneqq (\{\min M\}, 1)$
27: **else if** $t = 0$ **then**
28: Find the smallest number k satisfying $k \notin M$ and set $L[v] \coloneqq (\{k\}, 1)$.
29: **else if** $\alpha(v) = \max$ **then**
30: **if** $t = 1$ **then**
31: Set $L[v] \coloneqq (\{\max M\}, 1)$.
32: **else if** $t = 0$ **then**
33: Set $L[v] \coloneqq (\emptyset, 1)$.
34: Now let $(M, t) = L[v_C]$
35: **if** ($b \in M$ and $t = 1$) or ($b \notin M$ and $t = 0$) **then**
36: Return *true*, as $b \in I(v_C)$.
37: **else**
38: Return $false$.

■

4.2 Intersection with min and max

The problem MC($\cap, \min, \max$) is going to turn out to be in NL. Before we start, consider the following Lemma.

Lemma 4.3
Let $C \coloneqq (V, E, v_c, \alpha)$ be a $(\cap, \min, \max)$-gate. Further let $u \in V$ be an input gate and $v \in V$ be a gate such that v can be reached by u. Then either $I(v) = \emptyset$ or $I(v) = \{\alpha(u)\}$.

Proof. We show the statement inductively.

Initial case: For the input gate u we have $I(u) = \{\alpha(u)\}$.

Inductive step: Let $v \in V$ be a gate, which can be reached from u. Further let w resp. w_1, w_2 be the predecessor(s) of v.

(i) Let $\alpha(v) = \cap$. As v can be reached by u either w_1 or w_2 (or both) can be reached from u; w.l.o.g. let w_1 be the gate, which can be reached from u. Then either

$$I(v) = I(w_1) \cap I(w_2) = \{\alpha(u)\} \cap I(w_2) = \begin{cases} \{\alpha(u)\}, & \text{if } \alpha(u) \in I(w_2) \\ \emptyset, & \text{else} \end{cases}$$

or

$$I(v) = I(w_1) \cap I(w_2) = \emptyset \cap I(w_2) = \emptyset.$$

So either $I(v) = \{\alpha(u)\}$ or $I(v) = \emptyset$.

(ii) If $\alpha(v) = \min$, then

$$I(v) = \min I(w) = \begin{cases} \{\alpha(u)\}, & \text{if } I(w) = \{\alpha(u)\} \\ \emptyset, & \text{if } I(w) = \emptyset \end{cases}.$$

So again either $I(v) = \{\alpha(u)\}$ or $I(v) = \emptyset$.

(iii) The case $\alpha(v) = \max$ is analogous to (ii).

■

Theorem 4.4

$$\mathrm{MC}(\cap, \min, \max) \in \mathrm{NL}$$

Proof. Instead of showing $\mathrm{MC}(\cap, \min, \max) \in \mathrm{NL}$ directly, we show that the complement of this problem is in NL, which yields $\mathrm{MC}(\cap, \min, \max) \in \mathrm{NL}$ by using the fact that NL = coNL. So we are searching for a nondeterministic algorithm, which yields for a given $(\cap, \min, \max)$-circuit $C \coloneqq (V, E, v_c, \alpha)$ and $b \in \mathbb{N}$ the answer *yes*, if $b \notin I(C)$ and *no*, if $b \in I(C)$.

Assume for the moment that $b \notin I(C)$. Then there has to be an input gate u with $\alpha(u) \neq b$ such that v_C can be reached from v. (Otherwise, every input gate v, which reaches v_C, satisfies $\alpha(v) = b$.[4] Then $b \in I(C)$, as on the one hand for a set $\{b\}$ the minimum resp. maximum is $b = \min\{b\} = \max\{b\}$. On the other hand intersecting $\{b\}$ with $\{b\}$ yields $\{b\}$, so $I(C) = \{b\}$.)

Now assume there is an input gate u with $\alpha(u) \neq b$, from which we can reach v_C. There are two cases to distinguish:

(i) v_C can not be reached by any other input gate or every other input gate v, from which we can reach v_C, satisfies $\alpha(v) = \alpha(u)$. Then $I(C) = \{\alpha(u)\}$ as seen above and $b \notin I(C)$.

(ii) There is another input gate v, satisfying $\alpha(v) \neq \alpha(u)$ and v_C can be reached from v. Applying Lemma 4.3 yields $I(v_C) = \{\alpha(v)\}$ or $I(v_C) = \emptyset$. However v_C can also be reached from u, so $I(v_C) = \{\alpha(u)\}$ or $I(v_C) = \emptyset$. As $\alpha(v) \neq \alpha(u)$, the only possibility is $I(v_C) = \emptyset$, so $b \notin I(C)$.

We conclude $b \notin I(C)$ if and only if there is an input gate u such that v_C can be reached from u and $\alpha(u) \neq b$. This property can be be easily tested by the following algorithm.

[4] The case that no input gate can reach v_C would contradict the fact that C is acyclic.

Algorithm 4.2 Solving the complement of MC($\cap, \min, \max$)

Input: $(\cap, \min, \max)$-circuit $C \coloneqq (V, E, v_C, \alpha)$ and $b \in \mathbb{N}$.

1: **for** every gate $u \in V$ with $\alpha(u) \neq b$ **do**
2: Initialize $v \coloneqq u$.
3: **while** $v \neq v_C$ **do**
4: If v has no successors, break the while-loop. This path is a dead end.
5: Set nondeterministicly each successor of v as the new v.
6: Return *true*, as we found a path from u to v_C.
7: Return $false$.

This algorithm runs in (nondeterministic) logarithmic space, the only space needed is for managing the for-loop and saving the current v in the while-loop. ■

Corollary 4.5

$$\mathrm{MF}(\cap, \min, \max) \in \mathrm{L}$$

Proof. Let $F \coloneqq (V, E, v_C, \alpha)$ be a $(\cap, \min, \max)$-formula and let $b \in \mathbb{N}$. As F is a $(\cup, \min, \max)$-circuit, too, Algorithm 4.2 can be used to decide the question, if $b \notin I(C)$. However, since F is a formula, there is no need for nondeterministicly guessing the successors of a given gate in line 5. Instead we can search for the unique successor, resulting in a deterministic (logarithmic space) algorithm. ■

4.3 Union with min and max

This case is quiet an interesting one, as we have to differentiate, whether we add both minimum- and maximum-gates, or if we add just one of these types of gates. If we just add min- or max-gates, the problem will be in NL, while adding both min- and max-gates, the membership problem will be P-complete regarding $\leq_m^{\log}$-reduction.

Theorem 4.6

$$\mathrm{MC}(\cup, \min) \in \mathrm{NL} \textit{ and } \mathrm{MC}(\cup, \max) \in \mathrm{NL}$$

Proof. Let $C \coloneqq (V, E, v_C, \alpha)$ be a $(\cup, \min)$-circuit and $b \in \mathbb{N}$. Then $b \in I(C)$ if and only if there exists a path in C from an input gate $v \in V$ satisfying $\alpha(v) = b$ to v_C and the following condition is met: For every min-gate w along this path, there can not be an input gate $u \in V$ with $\alpha(u) < b$ and w can be reached by u. Otherwise, we would have $I(w) = \{x\}$ with $x \leq \alpha(u)$ and "the value b is lost" along this path.

The following algorithm returns *true* if and only if such a path exists.

Algorithm 4.3 Solving MC($\cup, \min$)

Input: $(\cup, \min)$-circuit $C \coloneqq (V, E, v_C, \alpha)$ and $b \in \mathbb{N}$.

1: **for** every gate $v \in V$ with $\alpha(v) = b$ **do**
2: Initialize $w \coloneqq v$
3: **while** $w \neq v_C$ **do**
4: If w has no successors, break the while-loop. This is a dead end.
5: **if** $\alpha(w) = \min$ **then**
6: **for** every $u \in V$ with $\alpha(u) < b$ **do**
7: Test, if w can be reached by u.
8: If *yes*, break the outer for-loop and continue with the next v.
9: Set nondeterministicly each successor of w as the new w.
10: Return *true*.
11: Return $false$.

At line 7 the test if w can be reached by u can be implemented via the idea of the proof of the famous Immerman–Szelepcsényi theorem. This ensures us that if w can be reached, there is a calculation path that returns this answer, and if w can not be reached, there is a calculation path returning this answer. Continue the calculation only in these cases. If a calculation path returns no definitive answer drop the calculation at this point.

Besides managing two for-loops (starting in lines 1 and 6) and one while-loop (line 3), the algorithm has to solve a reachability problem in line 7. All of these things can be done in nondeterministic logarithmic space, yielding MC($\cup, \min$) $\in$ NL.

The algorithm for solving MC($\cup, \max$) is analogous, the only differences are in line 5 ($\alpha(w) = \max$) and in line 6, where we are looking for input gates satisfying $\alpha(u) > b$ instead of $\alpha(u) < b$, showing MC($\cup, \max$) $\in$ NL. ■

Now let us study the case MC($\cup, \min, \max$). To see the P-hardness of MC($\cup, \min, \max$), we consider the monotone circuit value problem (MCV), which asks for a given boolean circuit C with AND- and OR-gates, if the circuit C evaluates *true*.

Here we define boolean circuit with AND- and OR-gates similar to algorithmic circuits (see Definition 2.1). Let $C \coloneqq (V, E, v_c, \alpha)$, then C is a monotone boolean circuit, if (V, E) is an acyclic graph. Further for an input gate $v \in V$, we have $\alpha(v) \in \{true, false\}$. If $v \in V$ is a computational gate, then $\alpha(v) \in \{\text{AND}, \text{OR}\}$. The interpretation function I is defined inductively:

- $I(v) \coloneqq \alpha(v)$ for input gates $v \in V$.
- Let $v \in V$ be a computational gate and $u_1, u_2 \in V$ its predecessors. If $\alpha(v) = \text{AND}$, then $I(v) \coloneqq I(u_1) \wedge I(u_2)$. If $\alpha(v) = \text{OR}$, then $I(v) \coloneqq I(u_1) \vee I(u_2)$.

From that define the set

$$\text{MCV} \coloneqq \{C \mid C \text{ is a boolean circuit and } I(C) \text{ is } true\}.$$

Goldschlager has shown in [Gol77] that the set MCV is $\leq_m^{\log}$-complete for P.

Theorem 4.7

$$\text{MCV} \leq_m^{\log} \text{MC}(\cup, \min, \max),$$

so $\text{MC}(\cup, \min, \max)$ *is* P*-hard.*

Proof. Let $C \coloneqq (V, E, v_C, \alpha)$ be a boolean circuit with AND- and OR-gates. We translate C into an algortihmic circuit $\tilde{C} \coloneqq (\tilde{V}, \tilde{E}, \tilde{v}_{\tilde{C}}, \tilde{\alpha})$ in the following way:

- For input gates $v \in V$, if $\alpha(v) = true$, set $\tilde{\alpha}(v) \coloneqq 1$. Otherwise set $\tilde{\alpha}(v) \coloneqq 0$.
- For computational gates $v \in V$, construct a new gate v_{new} and add (v, v_{new}) to E as well as (v_{new}, w) for all predecessors w of v. Delete all edges (v, w) from E. In other words: Add the gate v_{new} to C by placing it right after v.

  If $\alpha(v) = \text{AND}$, set $\tilde{\alpha}(v_{new}) \coloneqq \min$, otherwise if $\alpha(v) = \text{OR}$, set $\tilde{\alpha}(v_{new}) \coloneqq \max$. Set $\alpha(v) = \cup$ in the end in both cases.
- Set $\tilde{v}_{\tilde{C}} \coloneqq v_{Cnew}$.

Let I be the interpretation function regarding C and $\tilde{I}$ the interpretation function regarding $\tilde{C}$. Then $C \in \text{MCV} \Leftrightarrow (\tilde{C}, 1) \in \text{MC}(\cup, \min, \max)$, which we show by proofing the equivalences

$$I(v) = true \Leftrightarrow \tilde{I}(v_{new}) = \{1\} \text{ and } I(v) = false \Leftrightarrow \tilde{I}(v_{new}) = \{0\} \tag{4.1}$$

for every $v \in V$ inductively.

Initial case: For input gates $v \in V$ (4.1) is true by construction.

Inductive step: Let $v \in V$ be a computational gate with predecessors u^1, u^2

If $\alpha(v) = \text{AND}$, then

$$\begin{aligned} I(v) = true &\Leftrightarrow I(u^1) = true \text{ and } I(u^2) = true \Leftrightarrow \tilde{I}(u^1_{new}) = \{1\} \text{ and } \tilde{I}(u^2_{new}) = \{1\} \\ &\Leftrightarrow \tilde{I}(v) = \{1\} \Leftrightarrow \tilde{I}(v_{new}) = \min \tilde{I}(v) = \{1\} \\ I(v) = false &\Leftrightarrow I(u^1) = false \text{ or } I(u^2) = false \Leftrightarrow \tilde{I}(u^1_{new}) = \{0\} \text{ or } \tilde{I}(u^2_{new}) = \{0\} \\ &\Leftrightarrow \tilde{I}(v) = \{0\} \text{ or } \tilde{I}(v) = \{0, 1\} \Leftrightarrow \tilde{I}(v_{new}) = \min \tilde{I}(v) = \{0\}. \end{aligned}$$

Analogously, if $\alpha(v) = \text{OR}$, then

$$\begin{aligned} I(v) = true &\Leftrightarrow I(u^1) = true \text{ or } I(u^2) = true \Leftrightarrow \tilde{I}(u^1_{new}) = \{1\} \text{ or } \tilde{I}(u^2_{new}) = \{1\} \\ &\Leftrightarrow \tilde{I}(v) = \{1\} \text{ or } \tilde{I}(v) = \{0, 1\} \Leftrightarrow \tilde{I}(v_{new}) = \max \tilde{I}(v) = \{1\} \\ I(v) = false &\Leftrightarrow I(u^1) = false \text{ and } I(u^2) = false \Leftrightarrow \tilde{I}(u^1_{new}) = \{0\} \text{ and } \tilde{I}(u^2_{new}) = \{0\} \\ &\Leftrightarrow \tilde{I}(v) = \{0\} \Leftrightarrow \tilde{I}(v_{new}) = \max \tilde{I}(v) = \{0\}. \end{aligned}$$

This reduction is realised by Algorithm 4.4 on the next page. Besides managing one for-loop, this algorithm has to store the value of the pointer p. This can be done in logarithmic space.

Algorithm 4.4 Reducing MCV to MC($\cup, \min, \max$)

Input: boolean circuit $C \coloneqq (V, E, v_C, \alpha)$ with AND- and OR-gates.

Let n be the greatest number present in the names
of the gates $v \in V$ and set $m \coloneqq n + 1$.
for every $v \in V$ **do**
    if $\alpha(v) = true$ **then**
        Write v, $\alpha(v) \coloneqq 1$ as well as all successors of v onto the output.
    else if $\alpha(v) = false$ **then**
        Write v, $\alpha(v) \coloneqq 0$ as well as all successors of v onto the output.
    else if $\alpha(v) \in \{\text{AND}, \text{OR}\}$ **then**
        Write v and $\alpha(v) \coloneqq \cup$ onto the output.
        Set a new gate $\tilde{v}$ as the only predecessor of v.
        if $\alpha(v) = \text{AND}$ **then**
            Start a new gate entry by writing $\tilde{v}$, $\alpha(\tilde{v}) \coloneqq \min$
            and all predecessors of v onto the output.
        else if $\alpha(v) = \text{OR}$ **then**
            Start a new gate entry by writing $\tilde{v}$, $\alpha(\tilde{v}) \coloneqq \max$
            and all predecessors of v onto the output.
    if $v = v_C$ **then**
        Store a pointer p towards $\tilde{v}$.
Write the name, which is pointed at by p, as the new output gate onto the output.

■

4.4 The case of only min-, max-gates

At first glance the problem MC($\min, \max$) might look equivalent to the Reachability problem of graphs, however, one can use the fact, that every min-, max-gate has exactly one predecessor to solve MC($\min, \max$) - getting rid of the need for nondeterministicly guessing predecessor gates.

Theorem 4.8

$$\text{MC}(\min, \max) \in \text{L}$$

Proof. We have to find a logspace algorithm for deciding, if for a given ($\min, \max$)-circuit $C \coloneqq (V, E, v_C, \alpha)$ and $b \in \mathbb{N}$ the statement $b \in I(C)$ is true. Consider the following idea: We start at the output gate and work our way through the circuit backwards by taking the unique predecessors. In the end, we have to reach an input gate v. If $\alpha(v) = b$, then $I(C) = \{b\}$, as the min-, max-gates do not alter the values at all. (For an arbitrary set $N \coloneqq \{n\}$, $n = \min N = \max N$.) This yields the following algorithm.

Algorithm 4.5 Solving MC(min, max)

Input: (min, max)-circuit $C \coloneqq (V, E, v_C, \alpha)$ and $b \in \mathbb{N}$.

```
1: Initialize v ≔ v_C
2: while α(v) ∉ ℕ do
3:     Find the unique u with (u, v) ∈ E and set v ≔ u.
4: if α(v) = b then
5:     Return true.
6: else if α(v) ≠ b then
7:     Return false.
```

The memory space needed to perform this algorithm is saving the variable v. This can be realized in logarithmic space by using a pointer. ■

The same idea can be applied to formulas without any restrictions.

Corollary 4.9

$$\mathrm{MF}(\min, \max) \in \mathrm{L}$$

Proof. For solving MF(min, max) one can use Algorithm 4.5 once again. ■

5 Problems with both $+$ and $\times$

The question, if MC$(\cup, \cap, \overline{\ }, +, \times)$ is decidable, is an open question. McKenzie & Wagner showed, based on an idea first found by Christian Glaßer, that an algorithm for solving MC$(\cup, \cap, \overline{\ }, +, \times)$ could decide, whether the famous (unsolved) Goldbach Conjecture is true or false. The Conjecture states: *Every even number greater than 2 is the sum of two primes.*

Consider the formula

$$GE2 = \overline{(0 \cup 1)}$$

for the set of the numbers ≥ 2 and the formula

$$PRIMES = GE2 \cap (\overline{GE2 \times GE2})$$

for the set of all primes. Now contruct the formula

$$GOLDBACH = (GE2 \times 2) \cap (\overline{PRIMES + PRIMES}).$$

Then $I(GOLDBACH)$ is empty if and only if every even number greater than 2 is the sum of two primes, so

$$0 \in I\left(\overline{0 \times GOLDBACH}\right)$$

if and only if Goldbach's Conjecture is true.

By adding the option of having min-, max-gates, we can find even more open problems, which can be represented by a $(\cap, \cup, \overline{\ }, +, \times, \min, \max)$-formula. A good place to find "easy to formulate but hard to solve"-problems are problems about prime numbers. For example consider the following formula

$$TWINS = PRIMES \cap (PRIMES + 2)$$

for the set of twin primes. (More correctly, the set contains the greater prime of all twin prime pairs.) The open question, if there are infinitely many prime twins, is true if and only if $\max(TWINS)$ is empty. In the same way as above, we formulate the equivalent membership problem

$$0 \in I\left(\overline{0 \times \max TWINS}\right).$$

For the next example consider

$$SPRIMES = PRIMES \cap ((2 \times PRIMES) + 1)$$

the set of all safe primes, associated to a Sophie-Germain prime. (A prime p is called Sophie-Germain prime, if $2p+1$ is also prime. $2p+1$ is called a safe prime associated with p.) Then

$$0 \in I\left(\overline{0 \times \max SPRIMES}\right)$$

if and only if there are infinitely many Sophie-Germain primes, which is also an open question. More general, problems of the form: "Are there infinitely many numbers satisfying some condition A?" can be easily represented by finding a formula F_A describing the set of numbers satisfying condition A and then asking

$$0 \in I\left(\overline{0 \times \max F_A}\right),$$

since $\max F_A$ is empty if and only if, there are infinitely many numbers satisfying condition A. The tricky part however may be to find a formula F_A to describe condition A. At this point I encourage the reader to find more open problems, which can be represented by a $(\cap, \cup, {}^{-}, +, \times, \min, \max)$-formula.

Coming back to actually solving the membership problem, in this chapter, we are studying the problems, where both $+$ and $\times$ are possible computational gates. In section 5.1 we take a look at how to eliminate all min- max-gates efficiently from a $(\cap, +, \times, \min, \max)$-circuit reducing it to a circuit without any min-, max-gates. Similarly in section 5.2 we present a process of eliminating min-, max-gates of a $(\cup, +, \times, \min, \max)$-circuit. In section 5.3 we consider $(\cup, \cap, +, \times)$-formulas, the related membership problem turns out to be in Δ_2^{P}, a complexity class part of the polynomial-time hierarchy. From that we can conclude in section 5.4 that $\mathrm{MC}(\cup, \cap, +, \times) \in \Delta_2^{\mathrm{EXP}}$, the exponential-time analogy to Δ_2^{P}.

5.1 The case $(\cap, +, \times, \min, \max)$ and its subcases

Let $\mathcal{O} \subseteq \{\cap, +, \times\}$ with $\mathcal{O} \cap \{+, \times\} \neq \emptyset$ and $\mathcal{M} \coloneqq \{\min, \max\}$. In this section we will show $\mathrm{MC}(\mathcal{O} \cup \mathcal{M}) \leq_m^{\log} \mathrm{MC}(\mathcal{O})$. We will quickly go through the main idea of this reduction, the claims used here will be shown later in detail.

Let $C \coloneqq (V, E, v_C, \alpha)$ be a $(\mathcal{O} \cup \mathcal{M})$-circuit. Notice that for such a circuit C we have for every gate $v \in V$ that $|I(v)| \leq 1$. Applying a min- or max-gate on a set consisting of at most one element does not change the incoming set of its predecessor. This motivates the idea of substituting the min-, max-gates by either adding 0 or multiplying with 1, both of these operations do not change the incoming set, like the min-, max-gates before, resulting in an equivalent circuit.

Theorem 5.1
Let $\mathcal{O} \subseteq \{\cap, +, \times\}$ with $\mathcal{O} \cap \{+, \times\} \neq \emptyset$ and $\mathcal{M} \coloneqq \{\min, \max\}$. Then

$$\mathrm{MC}\left(\mathcal{O} \cup \mathcal{M}\right) \leq_m^{\log} \mathrm{MC}\left(\mathcal{O}\right)$$

Proof. Let $C \coloneqq (V, E, v_C, \alpha)$ be a $\mathcal{O} \cup \mathcal{M}$-circuit. First we will show $|I(v)| \leq 1$ for all $v \in V$ by induction.

Base Case: For all input gates $v \in V$ we have $|I(v)| = |\{\alpha(v)\}| \leq 1$.

Inductive step: Let $v \in V$ be a computational gate and let u resp. u_1, u_2 be the predecessors of v.

(i) If $\alpha(v) = \cap$, then

$$|I(v)| = \left\{ \begin{array}{ll} 1, & \text{if } I(u_1) = I(u_2) \neq \emptyset \\ 0, & \text{otherwise} \end{array} \right\} \leq 1$$

(ii) If $\alpha(v) \in \{+, \times\}$, then

$$|I(v)| = \left\{ \begin{array}{ll} 1, & \text{if } |I(u_1)| = |I(u_2)| = 1 \\ 0, & \text{otherwise} \end{array} \right\} \leq 1$$

(iii) If $\alpha(v) \in \{\min, \max\}$, then $|I(v)| \leq 1$ follows directly from the definition of min-, max-gates.

So $|I(v)| \leq 1$ for all $v \in V$ is indeed true, which directly impacts the values at min-, max-gates, let $v \in V$ with $\alpha(v) \in \{\min, \max\}$ and $u \in V$ its predecessor. Consider the two cases:

(i) If $I(u) = \{n\}$. Then

$$I(v) = \left\{ \begin{array}{ll} \min I(u), & \text{if } \alpha(v) = \min \\ \max I(u), & \text{if } \alpha(v) = \max \end{array} \right\} = \{n\} = I(u).$$

(ii) If $I(v) = \emptyset$. Then

$$I(v) = \left\{ \begin{array}{ll} \min I(u), & \text{if } \alpha(v) = \min \\ \max I(u), & \text{if } \alpha(v) = \max \end{array} \right\} = \emptyset = I(u).$$

We conclude $I(v) = I(u)$ holds for all $v \in V$ with $\alpha(v) \in \{\min, \max\}$. The reduction algorithm will eliminate these gates by substituting the min- resp. max-label by a $+$-label and adding a 0 input gate as its second predecessor. (If $+ \notin \mathcal{O}$, one can substitute by a $\times$-label and multiplying with a 1 input gate.

Algorithm 5.1 Reducing $\mathrm{MC}(\mathcal{O} \cup \mathcal{M})$ to $\mathrm{MC}(\mathcal{O})$ for $\mathcal{O} \subseteq \{\cap, +, \times\}$, $\mathcal{M} = \{\min, \max\}$, assuming $+ \in \mathcal{O}$.

Input: Circuit $C \coloneqq (V, E, g_C, \alpha)$

1: **for** every $v \in V$ **do**
2: **if** $\alpha(v) \notin \{\min, \max\}$ **then**
3: Write v, $\alpha(v)$ as well as all successors of v onto the output.
4: In this case, we do not alter the circuit at all.
5: **else if** $\alpha(v) \in \{\min, \max\}$ **then**
6: Write a new input gate $\tilde{v}, \alpha(\tilde{v}) \coloneqq 0, v$ onto the output.
7: Write $v, \alpha(v) \coloneqq +$ as well as all successors of v onto the output.

Analyzing the needed memory space, we need to manage one for-loop, starting in line 3, which we can do in logspace. We conclude $\mathrm{MC}(\mathcal{O} \cup \mathcal{M}) \leq_m^{\log} \mathrm{MC}(\mathcal{O})$.

■

Corollary 5.2
Let $\mathcal{O} \subseteq \{\cap, +, \times\}$ *with* $\mathcal{O} \cap \{+, \times\} \neq \emptyset$ *and* $\mathcal{M} \coloneqq \{\min, \max\}$. *Then*

$$\mathrm{MF}\left(\mathcal{O} \cup \mathcal{M}\right) \leq_m^{\log} \mathrm{MF}\left(\mathcal{O}\right)$$

Proof. Algorithm 5.1 serves as a reduction algorithm again, because if the input circuit C is a formula, the output generated by this algorithm will be a formula, too. ■

5.2 The case $(\cup, +, \times, \min, \max)$ and its subcases

Let $\mathcal{O} \subseteq \{\cup, +, \times\}$ and $\mathcal{M} \coloneqq \{\min, \max\}$. We will show $\mathrm{MC}(\mathcal{O} \cup \mathcal{M}) \leq_m^{\mathrm{P}} \mathrm{MC}(\mathcal{O})$ in this section.

Observation 5.3. Let $\mathcal{O} \subseteq \{\cup, +, \times\}$, $\mathcal{M} \coloneqq \{\min, \max\}$ and let $C \coloneqq (V, E, v_C, \alpha)$ be a $(\mathcal{O} \cup \mathcal{M})$-circuit. Consider the following observations for any computational gate $v \in V$ with predecessor(s) u resp. u_1, u_2.

(i) If $\alpha(v) = \cup$, then

$$\begin{aligned} \min I(v) &= \min\{\min I(u_1), \min I(u_2)\} \text{ and} \\ \max I(v) &= \max\{\max I(u_1), \max I(u_2)\} \end{aligned}$$

(ii) If $\alpha(v) = +$, then

$$\begin{aligned} \min I(v) &= \min I(u_1) + \min I(u_2) \text{ and} \\ \max I(v) &= \max I(u_1) + \max I(u_2) \end{aligned}$$

(iii) If $\alpha(v) = \times$, then

$$\begin{aligned} \min I(v) &= \min I(u_1) \cdot \min I(u_2) \text{ and} \\ \max I(v) &= \max I(u_1) \cdot \max I(u_2) \end{aligned}$$

(iv) If $\alpha(v) \in \{\min, \max\}$, then

$$\min I(v) = \max I(v) = \alpha(v) I(u)$$

From that we conclude that the minimum of $I(v)$ (resp. the maximum) is merely dependent on the minima (resp. maxima) of the predecessor(s) of v. This yields a first approach for a possible reduction algorithm:

1.) Initialize two lists $m_{\min}$ and $m_{\max}$. These will fulfill the condition $\min I(v) = m_{\min}[v]$ (resp. $\max I(v) = m_{\max}[v]$).

2.) For input gates $v \in V$ set $m_{\min}[v] \coloneqq m_{\max}[v] \coloneqq \alpha(v)$.

3.) Compute for computational gates $v \in V$ the entries $m_{\min}[v]$ and $m_{\max}[v]$ according to the rules stated in the above observations.

4.) Read through the computational $v \in V$ gates and substitute the min-gates by an input gate with the value at $m_{\min}[v]$ (resp. the max-gates by $m_{\max}[v]$).

While this is indeed a correct reduction, one can see based on the example below that the lengths of $m_{\min}$ resp. $m_{\max}$ can grow exponentially - destroying any hope of this being a polynomial time reduction.

Example 5.4. (See Figure 5.1.) For $n \in \mathbb{N}^+$ consider $C_n := (V_n, E_n, v_{out}, \alpha_n)$ with gates

$$V_n := \{v_{i,j} \mid 1 \leq i \leq j \leq n\} \cup \{v_{out}\},$$

edges

$$E_n := \{(v_{1,1}, v_{out})\} \cup \bigcup_{j=1}^{n-1} \bigcup_{i=1}^{j} \{(v_{i,j+1}, v_{i,j}), (v_{i+1,j+1}, v_{i,j})\}$$

and labels $\alpha(v_{out}) := \max$ and

$$\alpha(v_{i,j}) := \begin{cases} 2, & \text{if } j = n \\ \times, & \text{otherwise} \end{cases}.$$

The values of the interpretation function I are given by

$$I(v_{i,j}) = \{2^{2^{n-j}}\},$$

so the above stated reduction algorithm would need to calculate for the output gate

$$m_{\max}[v_{out}] = 2^{2^{n-1}},$$

which has length 2^{n-1}. Let us count the size of the input: We have at most $n^2 + 1$-many gates. Every gate has at most two gates as predecessors, which means the edges are sparse enough to neglect them. Giving $n^2 + 1$-many gates different names can be done with names, which have length approximately $\log(n^2 + 1)$. Putting everything together yields the input size is $O(n^2 \cdot \log(n^2))$. There is no polynomial p such that $p(n^2 \cdot \log(n^2)) \leq 2^{n-1}$ for all $n \in \mathbb{N}$, so the above stated reduction algorithm is not a polynomial time algorithm.

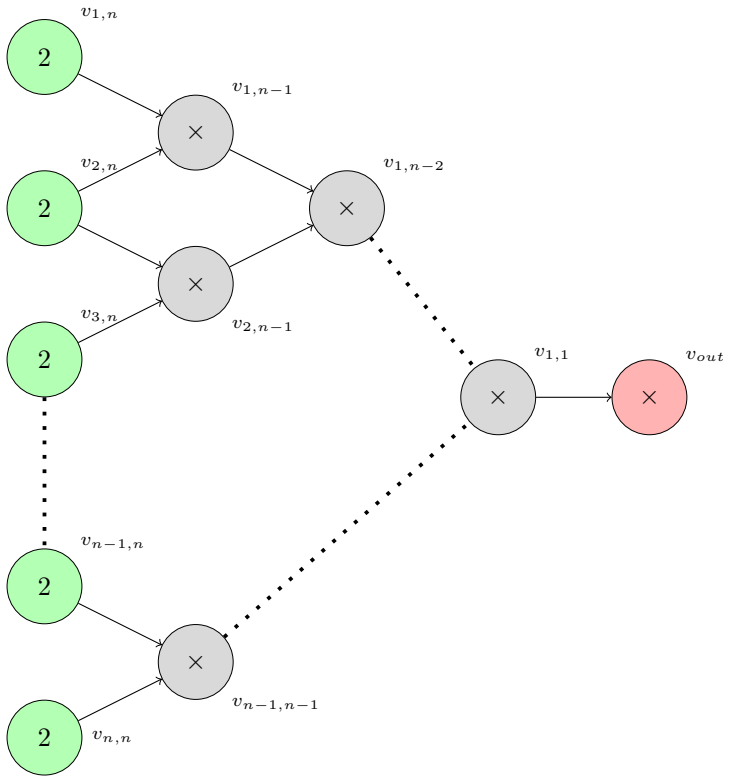


Figure 5.1: Illustration of the circuit C_n of Example 5.4

However, it is not necessary to calculate the minima (resp. maxima) explicitly, if those values are *too high*. In fact, when deciding whether $b \in I(C)$, it is sufficient to calculate the minima (resp. maxima) up to b. Once the values are greater than b, one can simply set these values to $b+1$. This is sufficient, because $\{\cup, +, \times, \min, \max\}$-gates are kind of *monotonous*: Once a certain threshold is reached, one can never get below that threshold again[1]. This will be proven in Lemma 5.6, but before we come to that, we will proof another Lemma, which will come in handy later on.

Lemma 5.5
Let $\mathcal{O} \subseteq \{\cup, +, \times\}$, $\mathcal{M} \coloneqq \{\min, \max\}$ and $C \coloneqq (V, E, v_C, \alpha)$ be a $(\mathcal{O} \cup \mathcal{M})$-circuit. Then $I(v)$ is finite and non-empty for all $v \in V$

Proof. We will show the statement inductively.

Base Case: Let $v \in V$ be an input gate. Then $|I(v)| = 1$, so $I(v)$ is finite and nonempty.

[1] There is one exception to this: Multiplying with 0. However, multiplying something with 0 gives 0 as solution, so the actual value we are multiplying with 0 is not relevant.

Inductive Step: Let $v \in V$ be a computational gate with predecessor(s) u resp. u_1, u_2. Then $I(u)$ resp. $I(u_1)$, $I(u_2)$ are finite and nonempty by the induction hypothesis.

(i) If $\alpha(v) = \cup$, we have

$$\max\{|I(u_1)|, |I(u_2)|\} \leq |I(v)| \leq |I(u_1)| + |I(u_2)|.$$

(ii) If $\alpha(v) \in \{+, \times\}$, we have

$$\min\{|I(u_1)|, |I(u_2)|\} \leq |I(v)| \leq |I(u_1)| \cdot |I(u_2)|.$$

(iii) If $\alpha(v) \in \{\min, \max\}$, then $|I(v)| = 1$, as both the minimum and the maximum of finite, nonempty subsets of $\mathbb{N}$ exists. So $I(v)$ is finite and nonempty.

In all cases $I(v)$ is finite and nonempty. ■

Lemma 5.6
Let $\mathcal{O} \subseteq \{\cup, +, \times\}$, $\mathcal{M} := \{\min, \max\}$, $b \in \mathbb{N}$ and $C := (V, E, v_C, \alpha)$ be a $(\mathcal{O} \cup \mathcal{M})$-circuit. Further let $w \in V$ with $\alpha(w) \in \{\min, \max\}$, then there is exists $n \in \mathbb{N}$ such that $I(w) = \{n\}$ (by Lemma 5.5). Construct from C a new circuit $\tilde{C} := (\tilde{V}, \tilde{E}, \tilde{v}_{\tilde{C}}, \tilde{\alpha})$ in the following way:

- $\tilde{V} := V$ *and* $\tilde{v}_{\tilde{C}} := v_C$.
- $\tilde{E} := E \setminus \{(u, w) \mid (u, w) \in E\}$, *i.e. if u is a predecessor of w, we delete that edge.*
- $\tilde{\alpha}(v) := \begin{cases} \alpha(v), & \text{if } v \neq w \\ n, & \text{if } v = w \text{ and } n \leq b \\ b+1, & \text{if } v = w \text{ and } n > b \end{cases}$

Then the equivalence

$$b \in I(C) \Leftrightarrow b \in I(\tilde{C})$$

is true.

Proof. In order to clarify things, we denote by I the interpretation function regarding the circuit C and $\tilde{I}$ the interpretation function regarding the circuit $\tilde{C}$. We verify the equivalence by proofing that for all $v \in V$ one of the following statements is true.

$$\begin{aligned} &(i) \quad I(v) = \tilde{I}(v). \\ &(ii) \quad I(v)_{\leq b} = \tilde{I}(v)_{\leq b} \text{ and there exist } x, \tilde{x} \in \mathbb{N} \\ &\qquad \text{with } x, \tilde{x} > b \text{ and } x \in I(v), \tilde{x} \in \tilde{I}(v), \end{aligned} \tag{5.1}$$

where for a set $M \subseteq \mathbb{N}$ we define $M_{\leq b} := M \cap \{0, 1, \ldots, b\}$. First, assume $v = w$, so $\alpha(v) \in \{\min, \max\}$ and $I(v) = \{n\}$ for some $n \in \mathbb{N}$ as the existence of such a Minimum

resp. Maximum is guaranteed by Lemma 5.5. If $n \leq b$, we have $I(v) = \{n\} = \tilde{I}(v)$, so (i) of (5.1) is fullfilled.

On the other hand, if $n > b$, we have $I(v) = \{n\}$ and $\tilde{I}(v) = \{b+1\}$. We conclude $I(v)_{\leq b} = \emptyset = \tilde{I}(v)_{\leq b}$ and in both $I(v)$ and $\tilde{I}(v)$ there is an element greater than b, so (ii) of (5.1) is fulfilled.

For $v \in V \setminus \{w\}$, we will show (5.1) inductively.

Base Case: Let $v \in V \setminus \{w\}$ be an input gate. Then $I(v) = \tilde{I}(v)$, as the construction of $\tilde{C}$ does not alter the input gates of C at all.

Inductive step: Let $v \in V \setminus \{w\}$ be a computational gate with predecessor(s) u resp. u_1, u_2. First assume via the induction hypothesis (i) of (5.1) is true for u resp. for both u_1 and u_2. As $\alpha(v) = \tilde{\alpha}(v)$, we conclude $I(v) = \tilde{I}(v)$, because applying the same operator on the same sets will always lead to the same solution sets, so again (i) of (5.1) is satisfied. From now on assume u resp. one of u_1 or u_2 satisfy (ii) of (5.1). Without loss of generality let this be u_1.

Let's investigate the different cases regarding the possible labels for v.

(i) If $\alpha(v) = \cup$, then

$$\begin{aligned} I(v)_{\leq b} &= (I(u_1) \cup I(u_2))_{\leq b} = I(u_1)_{\leq b} \cup I(u_2)_{\leq b} \\ &= \tilde{I}(u_1)_{\leq b} \cup \tilde{I}(u_2)_{\leq b} = (\tilde{I}(u_1) \cup \tilde{I}(u_2))_{\leq b} \\ &= \tilde{I}(v)_{\leq b}, \end{aligned}$$

so the first part of (ii) in (5.1) is true. As u_1 satisfies (ii), there exist $x \in I(u_1), \tilde{x} \in \tilde{I}(u_1)$ with $x, \tilde{x} > b$. As $I(v) = I(u_1) \cup I(u_2)$ and $\tilde{I}(v) = \tilde{I}(u_1) \cup \tilde{I}(u_2)$, we have $x \in I(v)$ and $\tilde{x} \in \tilde{I}(v)$, so the second part is also true.

(ii) Assume $\alpha(v) = +$ and let $n \in I(v)_{\leq b}$. Then there are $x \in I(u_1)$ and $y \in I(u_2)$ with $n = x + y$. By using the fact $(n = x + y \Rightarrow x, y \leq n)$, we get $x, y \leq b$, so $x \in I(u_1)_{\leq b} = \tilde{I}(u_1)_{\leq b}$ and $y \in I(u_1)_{\leq b} = \tilde{I}(u_2)_{\leq b}$. This means $x + y = n \in \tilde{I}(v)_{\leq b}$, hence $I(v)_{\leq b} \subseteq \tilde{I}(v)_{\leq b}$.

In order to show $I(v)_{\leq b} \supseteq \tilde{I}(v)_{\leq b}$ one can argue the exact same way, starting with $n \in \tilde{I}(v)_{\leq b}$ this time, which shows the first part of (ii) in (5.1).

Let $b < x \in I(u_1)$. In view of Lemma 5.5 $I(u_2)$ is nonempty, so there exists a $y \in I(u_2)$ and $x + y = z \in I(v)$ with $z > b$. With the same argument, one gets $b < \tilde{x} \in \tilde{I}(u_1)$, $\tilde{y} \in \tilde{I}(u_2)$ and $\tilde{x} + \tilde{y} = \tilde{z} \in \tilde{I}(v)$ with $\tilde{z} > b$, which shows the second part of (ii).

(iii) Now let $\alpha(v) = \times$. First assume $I(u_2) = \{0\}$. Then we have $\tilde{I}(u_2) = \{0\}$ and $I(v) = \tilde{I}(v) = \{0\}$ satisfying (i) of (5.1).

Next let $n \in I(v)_{\leq b}$.

(iii.i) First assume $n \neq 0$. Following the same line of argumentation as in the case $\alpha(v) = +$, we get the existence of $x \in I(u_1)_{\leq b} = \tilde{I}(u_1)_{\leq b}$ and $y \in I(u_2)_{\leq b} = \tilde{I}(u_2)_{\leq b}$ with $x \cdot y = n \in \tilde{I}(v)_{\leq b}$.

(iii.ii) Secondly assume $n = 0$. As $(0 = x \cdot y \Rightarrow x = 0 \vee y = 0)$, we know either $0 \in I(u_1)$ or $0 \in I(u_2)$. Without loss of generality assume $0 \in I(u_1)$. Using again Lemma 5.5, we get $\tilde{I}(u_2)$ is a nonempty set, so there exists a $z \in \tilde{I}(u_2)$. This means $0 \cdot z = 0 \in \tilde{I}(v)$. Since $0 \leq b$ for all $b \in \mathbb{N}$, we conclude $0 \in \tilde{I}(v)_{\leq b}$.

Combining both cases yields $I(v)_{\leq b} \subseteq \tilde{I}(v)_{\leq b}$.

Again one can argue in the same way to show $I(v)_{\leq b} \supseteq \tilde{I}(v)_{\leq b}$, so the first part of (ii) in (5.1) is indeed true.

There exists a number $b < x \in I(u_1)$ and a $0 < y \in I(u_2)$ as $I(u_2)$ is again nonempty and the case $I(u_2) = \{0\}$ was already covered[2], so $b < x \cdot y = z \in I(v)$. For the same reason, there is a $\tilde{z} \in \tilde{I}(v)$, showing the second part of (ii). (Note that $I(u_2) = \{0\} \Leftrightarrow \tilde{I}(u_2) = \{0\}$.)

(iv) We move on to the case $\alpha(v) = \min$. As $I(u), \tilde{I}(u)$ are nonempty, we have $I(v) = \{n\}$ and $\tilde{I}(u) = \{\tilde{n}\}$ for some $n, \tilde{n} \in \mathbb{N}$ and recall that u satisfies (ii) of (5.1).

There are to cases to distinguish: First assume $n \leq b$. Then we conclude $n = \tilde{n}$ as the minima of two sets being identical up to a threshold b are the same as long as the minima of these sets are less or equal than b. So $I(v) = \{n\} = \tilde{I}(v)$ and (i) of (5.1) is satisfied.

The second case to consider is $n > b$. Assuming $\tilde{n} \leq b$ would contradict the fact that $I(u)_{\leq b} = \tilde{I}(u)_{\leq b}$. This yields $I(v)_{\leq b} = \emptyset = \tilde{I}(v)_{\leq b}$. In addition, we have $n \in I(v)$ and $\tilde{n} \in I(v)$ with both n and $\tilde{n}$ being greater than b. This yields (ii) of (5.1).

(v) The last case to consider is $\alpha(v) = \max$. Once again from Lemma 5.5 we know $I(u)$ and $\tilde{I}(u)$ are nonempty and finite, so the maximum of both sets exist. Considering that u satisfies (ii) of (5.1), we know these maxima are greater than b, so $I(v)_{\leq b} = \emptyset = \tilde{I}(v)_{\leq b}$ and in both sets there exist elements greater than b, namely the respectively maxima. So (ii) of (5.1) is satisfied for v.

■

Now we have all the tools at our disposal to proof the main statement of this section.

Theorem 5.7
Let $\mathcal{O} \subseteq \{\cup, +, \times\}$ *and* $\mathcal{M} \coloneqq \{\min, \max\}$. *Then*

$$\mathrm{MC}(\mathcal{O} \cup \mathcal{M}) \leq_m^{\mathrm{P}} \mathrm{MC}(\mathcal{O})$$

[2]The case $I(u_1) = \{0\}$ is not possible, as this would contradict the assumption u_1 satisfies (ii) of (5.1).

Proof. Let $C \coloneqq (V, E, v_C, \alpha)$ be a $(\mathcal{O} \cup \mathcal{M})$-circuit. This reduction is realized by the following algorithm.

Algorithm 5.2 Reducing $\mathrm{MC}(\mathcal{O} \cup \mathcal{M})$ to $\mathrm{MC}(\mathcal{O})$ for $\mathcal{O} \subseteq \{\cup, +, \times\}$ and $\mathcal{M} = \{\min, \max\}$.

Input: Circuit $C \coloneqq (V, E, g_C, \alpha)$ and $b \in \mathbb{N}$

Initialize two lists $m_{\min}$ and $m_{\max}$.
for every $v \in V$ **do**
  if $\alpha(v) \in \mathbb{N}$ **then**
    Set $m_{\min}[v] \coloneqq m_{\max}[v] \coloneqq \min\{\alpha(v), b+1\}$.
  else if $\alpha(v) = \cup$ **then**
    Find u_1, u_2 with $(u_1, v), (u_2, v) \in E$
    Set $m_{\min}[v] \coloneqq \min\{m_{\min}[u_1], m_{\min}[u_2]\}$.
    and $m_{\max}[v] \coloneqq \max\{m_{\max}[u_1], m_{\max}[u_2]\}$.
  else if $\alpha(v) = +$ **then**
    Find u_1, u_2 with $(u_1, v), (u_2, v) \in E$.
    Set $m_{\min}[v] \coloneqq \min\{m_{\min}[u_1] + m_{\min}[u_2], b+1\}$
    and $m_{\max}[v] \coloneqq \min\{m_{\max}[u_1] + m_{\max}[u_2], b+1\}$.
  else if $\alpha(v) = \times$ **then**
    Find u_1, u_2 with $(u_1, v), (u_2, v) \in E$.
    Set $m_{\min}[v] \coloneqq \min\{m_{\min}[u_1] \cdot m_{\min}[u_2], b+1\}$
    and $m_{\max}[v] \coloneqq \min\{m_{\max}[u_1] \cdot m_{\max}[u_2], b+1\}$.
  else if $\alpha(v) = \min$ **then**
    Find u with $(u, v) \in E$.
    Set $m_{\min}[v] \coloneqq m_{\max}[v] \coloneqq m_{\min}[u]$.
  else if $\alpha(v) = \max$ **then**
    Find u with $(u, v) \in E$.
    Set $m_{\min}[v] \coloneqq m_{\max}[v] \coloneqq m_{\max}[u]$.
for every $v \in V$ **do**
  if $\alpha(w) \in \{\min, \max\}$ **then**
    for every $w \in V$ with $(v, w) \in E$ **do**
      Delete $E \coloneqq E \setminus \{u, v\}$
  if $\alpha(v) = \min$ **then**
    Set $\alpha(v) \coloneqq m_{\min}[v]$.
  else if $\alpha(v) = \max$ **then**
    Set $\alpha(v) \coloneqq m_{\max}[v]$.
return Circuit C and b.

Applying Algorithm 5.2 on C does indeed eliminate all min- and max-gates, resulting in a $\mathcal{O}$-circuit. The correctness of this reduction algorithm, i.e. that $b \in I(C) \Leftrightarrow b \in I(\tilde{C})$ is given by Observation 5.3 and Lemma 5.6. The algorithm consists of two for-loops, each applying at most basic arithmetic operations on elements bounded by the size of b, which is a part of the input. Hence the algorithm is indeed a polynomial time reduction. ■

Corollary 5.8
Let $\mathcal{O} \subseteq \{\cup, +, \times\}$ *and* $\mathcal{M} \coloneqq \{\min, \max\}$. *Then*

$$\mathrm{MF}(\mathcal{O} \cup \mathcal{M}) \leq_m^{\mathrm{P}} \mathrm{MF}(\mathcal{M})$$

Proof. Let $b \in N$ and $F \coloneqq (V, E, v_C, \alpha)$ be a $(\mathcal{O} \cup \mathcal{M})$-formula. Applying Algorithm 5.2 on (F, b) gives an $\mathcal{O}$-circuit $\tilde{F}$. This circuit $\tilde{F}$ is in fact a formula, as Algorithm 5.2 does not add any edges to the input formula F while constructing $\tilde{F}$. So Algorithm 5.2 is a correct polynomial reduction in the context of formulas. ■

5.3 $(\cap, \cup, +, \times, \min, \max)$-formulas

Let $\mathcal{O} \coloneqq \{\cup, \cap, +, \times, \min, \max\}$, in this section we will show $\mathrm{MF}(\mathcal{O}) \in \Delta_2^P$. Regarding the lower bound, in section 6.2 we will see that $\mathrm{MF}(\cup, \cap, +, \min, \max)$ is Δ_2^{P}-hard, which immediately results in the Δ_2^{P}-hardness of $\mathrm{MF}(\mathcal{O})$.

The reader may be familiar with the complexity class Δ_2^{P}, which is part of the polynomial-time hierarchy, a hierarchy of complexity classes all at least as big as P but contained within PSPACE. In case the reader is not familiar with this concept, here is a quick definition of this complexity class.

Definition 5.9. Define the complexity class

$$\Delta_2^{\mathrm{P}} \coloneqq \mathrm{P}^{\mathrm{NP}},$$

meaning Δ_2^{P} consists of all problems solvable in polynomial time with access to any oracle B satisfying $B \in \mathrm{NP}$.

Let $\mathcal{O} \coloneqq \{\cup, \cap, +, \times, \min, \max\}$. The central idea for proofing $\mathrm{MF}(\mathcal{O}) \in \Delta_2^{\mathrm{P}}$ will be explained in the following. Let $b \in \mathbb{N}$ and $F \coloneqq (V, E, v_F, \alpha)$ be a $\mathcal{O}$-formula. We search for a polynomial time algorithm with access to NP-oracles, which decides the question, if $b \in I(F)$. This algorithm searches for the first min-, max-gate v, computes $I(v)$ explicitly by using (a modified version of) $\mathrm{MF}(\cup, \cap, +, \times)$ as an oracle and replaces the label of v with the computed value. (If $I(v)$ is empty, set the label to $\cap$, and set 0 and 1 as its predecessor.) After that, continue the same process for the next min-, max-gate until there are no more left. In the end, C is a $(\cup, \cap, +, \times)$-formula, the question if $b \in I(F)$ can now be solved by an $\mathrm{MF}(\cup, \cap, +, \times)$-oracle.

Before diving deeper into the details of this algorithm, let us establish an upper bound for the possible values of a $(\cup, \cap, +, \times, \min, \max)$-formula.

Lemma 5.10
Let $F \coloneqq (V, E, v_F, \alpha)$ *be a* $(\cup, \cap, +, \times, \min, \max)$*-formula and denote the set of all input gates by* $J \coloneqq \{v \in V \mid \alpha(v) \in \mathbb{N}\}$. *Then for all* $v \in V$ *and all* $x \in I(v)$ *the inequality*

$$x \leq \prod_{j \in J} (\alpha(j) + 1)$$

is true. For the remainder of this section denote by N_F this product for a given formula F.

Proof. McKenzie and Wagner stated in the proof of Lemma 4.1 in [MW07] N_F as an upper bound for $(\cup, \cap, +, \times)$-formulas. Further notice that for every $v \in V$ the set $I(v)$ is finite. The minimum resp. maximum of a finite set $I(v)$ is always a value already present in $I(v)$, so adding the option of having min-, max-gates can not produce any new values. We conclude the stated upper bound holds for the values of a $(\cup, \cap, +, \times, \min, \max)$-formula, too. ■

Let us come back to our algorithm and assume for the moment, we want to eliminate a max-gate named v with predecessor w. Now that we established the upper bound N_F, one may tempted to eliminate v in the following way. Test for all $0 \leq k \leq N_F$, starting with N_F and going downwards from there, if $(F_w, k) \in \mathrm{MF}(\cup, \cap, +, \times)$ (remember Definition 2.5). From Table 3.1 we know $\mathrm{MF}(\cup, \cap, +, \times) \in \mathrm{NP}$, so we can use this set as an oracle. However the length of N_C is polynomially bounded in the length of the input, so there would be exponentially many numbers k to test. We have to find a smarter way.

Instead of testing for a single number k, it would be nice, if we could test for a whole range of numbers from l to u, $l, u \in \mathbb{N}$, if there is a $k \in [l, u]$, such that $(F_w, k) \in \mathrm{MF}(\cup, \cap, +, \times)$. Start with the range $l \coloneqq 0$ and $u \coloneqq N_F$ and test, if there is a $0 \leq k \leq N_F$ such that $(F_w, k) \in \mathrm{MF}(\cup, \cap, +, \times)$. If the answer is no, we know that $I(v) = \emptyset$, so replace v by a construction computing the empty set. If the answer is yes, test, if there is a ${}^{N_F}/_2 \leq k \leq N_F$ such that $(F_w, k) \in \mathrm{MF}(\cup, \cap, +, \times)$. If yes, set $l = {}^{N_F}/_2$, and if no, set to $u = {}^{N_F}/_2$. Continue searching in the new range $[l, u]$, until we have $l = u$. Then we know $I(v) = \{l\}$, and we can substitute the label of v by l. We are basically performing a *binary search* on the range $[0, N_F]$.

Consider the following *ranged version* of the membership problem of formulas.

Definition 5.11. Let $\mathcal{O} \subseteq \{\cup, \cap, \bar{}, +, \times, \min, \max\}$. Define the set

$$\mathrm{MF}_{range}(\mathcal{O}) \coloneqq \{(F, l, u) \mid \text{there exists } k \in [l, u] \text{ with } (F, k) \in \mathrm{MF}(\mathcal{O})\}$$

Lemma 5.12

$$\mathrm{MF}_{range}(\cup, \cap, +, \times) \in \mathrm{NP}$$

Proof. From Table 3.1 we know that $\mathrm{MF}(\cup, \cap, +, \times) \in \mathrm{NP}$. Using the definition of the class NP over polynomial projections, this means there exists a $B \in \mathrm{P}$ and a polynomial p satisfying

$$\mathrm{MF}(\cup, \cap, +, \times) = \{(F, b) \mid \exists z \text{ with } |z| \leq p(|(F, b)|) \text{ and } (F, b, z) \in B\}. \tag{5.2}$$

Without loss of generality let p be monotonically increasing. Based on B define a new set

$$B_{range} \coloneqq \{(F, l, u, k, z) \mid l \leq k \leq u \text{ and } (F, k, z) \in B\}.$$

$B_{range} \in \mathrm{P}$, as one has to test, if $l \leq k \leq u$ and $(F, k, z) \in B$, both of which are possible in polynomial time. Further define the polynomial $q(x) \coloneqq \mathrm{id}(x) + p(x)$. Then, if

$$\begin{aligned}\mathrm{MF}_{range}(\cup, \cap, +, \times) = \{(F, l, u) \mid &\exists (k, z) \text{ with } |(k, z)| \leq q(|(F, l, u)|) \\ &\text{and } (F, l, u, k, z) \in B_{range}\} =: A,\end{aligned}$$

is true, we show $\mathrm{MF}_{range}(\cup, \cap, +, \times)$ satisfies the definition of NP-sets. First assume $(F, l, u) \in \mathrm{MF}_{range}(\cup, \cap, +, \times)$. This means there exists $k \in [l, u]$ such that $(F, k) \in \mathrm{MF}(\cup, \cap, +, \times)$. (5.2) implies there exists some z satisfying $|z| \leq p(|(F, k)|)$ and $(F, k, z) \in B$. This yields $(F, l, u, k, z) \in B_{range}$. For the length of (k, z), compute

$$|(k, z)| = |k| + |z| \leq |u| + p(|(F, k)|) \leq |(F, l, u)| + p(|(F, l, u)|) \leq q(|(F, l, u)|),$$

where $p(|(F, k)|) \leq p(|(F, l, u)|)$ follows from the monotony of p. We conclude $(F, l, u) \in A$.

Now let $(F, l, u) \in A$. This implies the existence of (k, z) such that $(F, l, u, k, z) \in B_{range}$, so one can find $k \in [l, u]$ such that $(F, k, z) \in B$, so $k \in I(F)$. This yields $(F, l, u) \in \mathrm{MF}_{range}(\cup, \cap, +, \times)$. ■

Now we have all the needed tools at our disposal to solve $\mathrm{MF}(\cup, \cap, +, \times, \min, \max)$.

Theorem 5.13
Let $\mathcal{O} \subseteq \{\cup, \cap, +, \times\}$ *and* $\mathcal{M} \coloneqq \{\min, \max\}$. *Then*

$$\mathrm{MF}(\mathcal{O} \cup \mathcal{M}) \in \Delta_2^{\mathrm{P}}$$

Proof. Without loss of generality let $\mathcal{O} = \{\cup, \cap, +, \times\}$. Consider algorithm 5.3 on the next page.

Let $F \coloneqq (V, E, v_F, \alpha)$ and $b \in \mathbb{N}$. Further let $v \in V$ with $\alpha(v) \in \{\min, \max\}$ and predecessor w. If there is no $k \in [0, N_F]$ with $k \in I(w)$, we know from Lemma 5.10 that $I(w) = \emptyset = I(v)$, which is exactly what the algorithm constructs in this case. Now assume there is a $k \in [0, N_F]$ such that $k \in I(w)$, If $\alpha(v) = \min$ the algorithm searches for the smallest $k_{min} \in [0, N_F]$ satisfying $k_{min} \in I(w)$. Then we know $I(v) = \{k_{min}\}$, which the algorithm constructs at the gate v. Analogously if $\alpha(v) = \max$ we search for the greatest $k_{max} \in [0, N_F]$ satisfying $k_{max} \in I(w)$. In this case Lemma 5.10 ensures us, there can not be a $k > N_F$, such that $k \in I(w)$, so $I(v) = \{k_{max}\}$, which the algorithm constructs at v. In the end, there are no more min-, max-gates, so we can use the set $\mathrm{MF}(\cup, \cap, +, \times)$ as an oracle to solve the question, if $b \in I(F)$.

The upper bound N_F can be computed in polynomial time, as it is the product of factors numbers bounded in the size of the input. Then we perform a binary search in the range $[0, N_F]$, whose number of steps is bounded polynomially. The runtime of every step is polynomially bounded. Combining all this, results in a polynomially bounded runtime. We use the sets $\mathrm{MF}_{range}(\cup, \cap, +, \times)$ and $\mathrm{MF}(\cup, \cap, +, \times)$ as oracles, according to Lemma 5.12 and Table 3.1 these sets are in NP. We conclude $\mathrm{MF}(\cup, \cap, +, \times, \min, \max) \in \Delta_2^{\mathrm{P}}$.

Algorithm 5.3 Solving $\mathrm{MF}(\cup, \cap, +, \min, \max)$

Input: $(\cup, \cap, +, \times, \min, \max)$-formula $F = (V; E, v_F, \alpha)$ and $b \in \mathbb{N}$

Compute the upper bound $N_F \coloneqq \prod_{\text{input gates } v} \alpha(v) + 1$.
for every $v \in V$ **do**
    if $\alpha(v) \in \{\min, \max\}$ **then**
        Initialize $l \coloneqq 0$, $u \coloneqq N_F$ and let w be the predecessor of v.
        if $(F_w, l, u) \notin \mathrm{MF}_{range}(\cup, \cap, +, \times)$ **then**
            Add two new gates v_1, v_2 to F and set $(v_1, v), (v_2, v) \in E$ as well as
            $\alpha(v_1) = 1, \alpha(v_2) = 0$. Overwrite $\alpha(v) = \cap$, find the edge $(u, v) \in E$
            and delete it. Break this for-loop and continue with the next v.
        if $\alpha(v) = \max$ **then**
            while $l \neq u$ **do**
                Set $m \coloneqq \lceil {}^{u+l}/_{2} \rceil$.
                if $(F_w, m, u) \in \mathrm{MF}_{range}(\cup, \cap, +, \times)$ **then**
                    Set $l \coloneqq m$.
                else
                    Set $u \coloneqq m - 1$.
        else if $\alpha(v) = \min$ **then**
            while $l \neq u$ **do**
                Set $m \coloneqq \lfloor {}^{u+l}/_{2} \rfloor$.
                if $(F_w, l, m) \in \mathrm{MF}_{range}(\cup, \cap, +, \times)$ **then**
                    Set $u \coloneqq m$.
                else
                    Set $l \coloneqq m + 1$.
        Find the edge $(u, v) \in E$ and delete it from E.
        Overwrite the label $\alpha(v) \coloneqq l$.
if $(F, b) \in \mathrm{MF}(\cup, \cap, +, \times)$ **then**
    Return *true*.
else
    Return *false*.

■

5.4 $(\cap, \cup, +, \times, \min, \max)$-circuits

Let $\mathcal{O} \subseteq \{\cup, \cap, +, \times\}$ and $\mathcal{M} \coloneqq \{\min, \max\}$. Wagner and McKenzie showed that $\mathrm{MC}(\mathcal{O}) \in$ NEXPTIME (Table 3.1) by arguing that a given $\mathcal{O}$-circuit C can be unfolded into a potentially exponential large formula, and then applying the algorithm for solving $\mathrm{MF}(\mathcal{O})$, which they showed to be NP-complete. We will argue in the same way, while providing a more in detail analysis of how a given circuit can be unfolded into an equivalent formula.

In analogy to the polynomial-time hierarchy, there exists the exponential-time hier-

archy. Solving a problem by exponentially unfolding the problem into a Δ_2^{P}-problem may unsurprisingly lead to the set complexity set Δ_2^{EXP}

Definition 5.14. Define the complexity class

$$\Delta_2^{\mathrm{EXP}} \coloneqq \mathrm{EXP}^{\mathrm{NP}},$$

meaning Δ_2^{EXP} consists of all problems solvable in exponential time with access to any oracle B satisfying $B \in \mathrm{NP}$.

So let $C \coloneqq (V_C, E_C, v_C, \alpha_C)$ be a $(\cap, \cup, +, \times, \min, \max)$-circuit and let $b \in \mathbb{N}$. Unfold this circuit into a formula $F \coloneqq (V_F, E_F, v_F, \alpha_F)$ in the following way. Consider an empty work tape. On this tape, we start writing a modified encoding of the unfolded formula F, which will not be topologically ordered (for the moment). Additionally, we create a new entry for every gate, in order to store the information, which gate it is associated to in the original circuit. We start the process by writing the output gate of C, giving it the new name 0 and ignoring any potential successors of v_C. We write:

$$0, v_C, \alpha(v_C)$$

Next, we look for the predecessor(s) of v_C. Assume, there are two predecessors u_1, u_2. They will get the new names 1 and 2, and we set 0 as their successor:

$$0, v_C, \alpha(v_C) - 1, u_1, \alpha(u_1), 0 - 2, u_2, \alpha(u_2), 0$$

After that we keep on jumping to the next gate on the work tape (at first the gate 1, then gate 2, etc.) and repeat the process of finding the predecessors of their associated gates in C and writing those predecessors down on the work tape as new entrys. In case there are no predecessors, i.e., it is in an input gate, do not write anything and jump immediately to the next gate. If there is no more next gate on the work tape, this process will terminate. (This is going to happen, as C is acyclic, so every *predecessor-chain* will run into an input gate at some point.)

In the end, reverse the ordering of the gates on the work tape and delete the second entry of every gate, i.e. the associated gate in C. Additionally mark 0 as the output gate. This will result in a correct encoding of a topologically ordered formula F, satisfying $I(C) = I(F)$.

In the end apply Algorithm 5.3 for solving $\mathrm{MF}(\cup, \cap, +, \times, \min, \max)$ with input (F, b) and return the same answer.

Before we describe the algorithm formally, we first introduce a Lemma, which we will need later to proof exponential runtime.

Definition 5.15. Let $C \coloneqq (V, E, v_C, \alpha)$ be a circuit. For the remainder of this section define for every gate $v \in V$

$$l(v) \coloneqq \text{the length of the longest path from } v \text{ to } v_C \text{ in } C,$$

Lemma 5.16
Let $C \coloneqq (V, E, v_C, \alpha)$ *be a* $(\cup, \cap, +, \times, \min, \max)$*-circuit and and define the set*

$$V(k) \coloneqq \{v \in V \mid l(v) \leq k\}.$$

Then

$$|V(k)| \leq 2^k.$$

Proof. We show the statement inductively.

Base Case: Consider $W(0)$. As C is acyclic, there is exactly one gate $v \in V$ satisfying $l(v) \leq 0$, namely $v = v_C$, so

$$|V(0)| = 1 \leq 2^1.$$

Inductive Step: Let $v \in V(k+1)$ and notice that all the successors $w_i \in V$ of v satisfy $w_i \in V(k)$, otherwise there would exist a longer path than $k+1$ from v to v_C. So the number of elements in $V(k+1)$ is bounded by the number of possible predecessors the gates in $V(k)$ can have.

By the induction hypothesis, there at most 2^k many possible successors of v. By definition of circuits, every gate has at most two predecessors, so there can be at most $2 \cdot 2^k$-many gates having one of the w_i as their predecessors, showing

$$|V(k+1)| \leq 2 \cdot 2^k = 2^{k+1}.$$

■

Theorem 5.17
Let $\mathcal{O} \subseteq \{\cup, \cap, +, \times\}$ *and* $\mathcal{M} \coloneqq \{\min, \max\}$*. Then*

$$\mathrm{MC}(\mathcal{O} \cup \mathcal{M}) \in \Delta_2^{\mathrm{EXP}}$$

Proof. Without loss of generality let $\mathcal{O} = \{\cup, \cap, +, \times\}$. Let $C \coloneqq (V, E, v_C, \alpha)$ be a $(\mathcal{O} \cup \mathcal{M})$-circuit and $b \in \mathbb{N}$. Algorithm 5.4 on the next page realizes the above stated construction of a $(\mathcal{O} \cup \mathcal{M})$-formula F satisfying $I(F) = I(C)$.

The runtime of every step in the while-loop starting in line 3 is polynomially bounded. But how many steps does the while-loop take? For that denote for every gate $v \in V$

$$n(v) \coloneqq \text{the number of gates associated to } v \text{ in } F.$$

We will show

$$n(v) \leq 2^{l(v)}$$

for all $v \in V$ inductively over the size of $l(v)$.

Base Case: Only $v = v_C$ satisfies $l(v) = 0$. The algorithm is initialized by writing a gate associated to v_C, so $n(v) \geq 1$. Further $n(v) > 1$ would contradict the fact that C is acyclic, so

$$n(v) = 1 = 2^0 = 2^{l(v)}.$$

Algorithm 5.4 Solving MC($\cap, \cup, +, \times, \min, \max$)

Input: $(\cap, \cup, +, \times, \min, \max)$-circuit $C := (V_C, E_C, v_C, \alpha_C)$ and $b \in \mathbb{N}$

Write $0, v_C, \alpha(v_C)$ onto an empty work tape.
Initialize two variables $m := 0$ and $n := 1$.
while $m \neq n$ **do**
    Find gate m on the work tape and look up its associated gate $v \in V_C$
    for every gate $u \in V_C$ with $(u, v) \in E_C$ **do**
        Create a new entry $n, u, \alpha(u), m$ on the work tape.
        Increment n by 1.
    Increment m by 1.
Reverse the ordering of the gates on the work tape, while deleting the second entry
of every gate and Set 0 as the output gate.
Name the formula on the work tape F, start Algorithm 5.3 with input (F, b)
and return the same answer.

Inductive step: Let $v \in V$ with $l(v) = k + 1$ and let $w_i \in V$ be the successors of v in C. Every gate of F associated to one of the w_i creates exactly one gate associated with v and other gates can not create gates associated to v, so

$$n(v) = \sum_i n(w_i).$$

As covered in the proof of Lemma 5.16, we know that $l(w_i) \leq k$, applying the induction hypothesis we get

$$n(v) = \sum_i n(w_i) \leq \sum_i 2^{l(w_i)} \leq \sum_i 2^k.$$

By Lemma 5.16 the number of successors w_i is bounded by 2^k, so

$$n(v) \leq \sum_i 2^k = 2^k \cdot 2^k = 2^{k+1} = 2^{l(v)},$$

which concludes the induction.

For every gate $v \in V$ the number $l(v)$ is bounded by $|V|$, as the longest path from v to v_C can not be longer than the number of gates present in C. This yields for the number of steps performed in the while-loop

$$\sum_{v \in V} n(v) \leq \sum_{v \in V} 2^{l(v)} \leq \sum_{v \in V} 2^{|V|} \leq |V| 2^{|V|}.$$

Additionally, we apply Algorithm 5.3 on an input, whose length is exponentially bounded. As seen in Theorem 5.13, Algorithm 5.3 has polynomially runtime, resulting in exponential runtime in total. Of Algorithm 5.3 uses the set $\mathrm{MF}_{range}(\cup, \cap, +, \times)$ as a NP-oracle, which have to take into account here also, which shows

$$\mathrm{MC}(\cup, \cap, +, \times, \min, \max) \in \mathrm{EXP}^{\mathrm{NP}}.$$

■

6 Problems with either + or $\times$

Let $\odot \in \{+, \times\}$, $\mathcal{O} \subseteq \{\cup, \cap, {}^{-}, \odot\}$ and $\mathcal{M} \coloneqq \{\min, \max\}$. In this chapter we will discuss problems of the form $\mathrm{MC}(\mathcal{O} \cup \mathcal{M})$ as well as their formula counterparts. A lot of cases, namely $\mathrm{MC}(\{\odot\} \cup \mathcal{M})$, $\mathrm{MC}(\{\cup, \odot\} \cup \mathcal{M})$ and $\mathrm{MC}(\{\cap, \odot\} \cup \mathcal{M})$, were already covered in the previous chapter as subcases of $\mathrm{MC}(\{\cup, +, \times\} \cup \mathcal{M})$ resp. $\mathrm{MC}(\{\cap, +, \times\} \cup \mathcal{M})$. (See Theorem 5.1 resp. Theorem 5.7.)

In section 6.1, we will show $\mathrm{MC}(\{\cup, \cap, {}^{-}, +\} \cup \mathcal{M}) \in \mathrm{PSPACE}$, which immediately yields $\mathrm{MC}(\{\cup, \cap, +\} \cup \mathcal{M}) \in \mathrm{PSPACE}$. In section 6.2 we will show Δ_2^{P} is a lower bound for $\mathrm{MF}(\{\cup, \cap, +\} \cup \mathcal{M})$, giving us a lower bound for $\mathrm{MF}(\cup, \cap, {}^{-}, +\}, \cup\mathcal{M})$, too. It would have been nice to conclude this chapter by solving $\mathrm{MC}(\{\cup, \cap, {}^{-}, \times\} \cup \mathcal{M})$ and their subcases, however, section 6.3 argues why that is quiet difficult.

6.1 The case of $(\cup, \cap, {}^{-}, +, \min, \max)$-circuits

The goal of this section is to show $\mathrm{MC}(\cup, \cap, {}^{-}, +, \min, \max) \in \mathrm{PSPACE}$ by explicitly finding a PSPACE-algorithm solving $\mathrm{MC}(\cup, \cap, {}^{-}, +, \min, \max)$. From McKenzie & Wagner (Table 3.1) we know there exists a PSPACE-algorithm for solving $\mathrm{MC}(\cup, \cap, {}^{-}, +)$; let us call that algorithm $\mathit{oracle}(C, b)$. Now let $b \in \mathbb{N}$ and $C \coloneqq (V, E, v_C, \alpha)$ be a $\{\cup, \cap, {}^{-}, +, \min, \max\}$-circuit. The idea now is to eliminate the min- resp. max-gates one-by-one.

First consider a min-gate $v \in V$ with predecessor $u \in V$. Then test whether $(C_u, i) \in \mathrm{MC}(\cup, \cap, {}^{-}, +)$ by calculating $\mathit{oracle}(C_u, i)$, where i starts at 0 and is incremented each time until the threshold $2^{|C|}$. (The reason why we use this threshold will be given in Lemma 6.2.) Once this test is successful, set v as an input gate with label i. If this test is never successful, we can be sure that $I(u) = \emptyset$ and we substitute the gate v by a construction computing the empty set.

For a max-gate test at first if $(C_u, 2^{|C|}+1) \in \mathrm{MC}(\cup, \cap, {}^{-}, +, \min, \max)$. If the answer is *yes*, we can conclude that $I(u)$ is infinite, we substitute v by a construction computing the empty set. If the answer is *no*, continue in the same way as described in the min-case, this time counting i down from $2^{|C|}$ to 0.

In the end we will have eliminated all min-, max-gates resulting in a $(\cup, \cap, {}^{-}, +)$-circuit. Compute $\mathit{oracle}(C, b)$ and return its answer.

Definition 6.1. Let $C \coloneqq (V, E, v_C, \alpha)$ be a circuit. For a gate $v \in V$ denote by $C(v)$ the subcircuit, which results from C by deleting all gates, from which v can not be reached.

Lemma 6.2
Let $C \coloneqq (V, E, v_C, \alpha)$ be a $(\cup, \cap, \overline{\ }, +, \min, \max)$-circuit. Then for every $v \in V$ either

$$I(v) \subseteq \{0, \dots, 2^{|C|}\} \text{ or } \overline{I(v)} \subseteq \{0, \dots, 2^{|C|}\}.$$

Proof. We will show the statement

$$I(v) \subseteq \{0, \dots, 2^{|C(v)|}\} \text{ or } \overline{I(v)} \subseteq \{0, \dots, 2^{|C(v)|}\}$$

inductively. As $|C(v)| \leq |C|$ for all $v \in V$, this proofs the lemma.

Base Case: Let $v \in V$ be an input gate. Then $I(v) = \{\alpha(v)\} = \{2^{\log(\alpha(v))}\}$. As $C(v)$ contains the encoding of the value $\alpha(v)$, we have $\log(\alpha(v)) \leq |C(v)|$, so $I(v) \subseteq \{0, \dots, 2^{|C(v)|}\}$.

Inductive step: Let $v \in V$ be a computational gate with predecessor(s) u resp. u_1, u_2 and assume the induction hypothesis is true for these predecessor(s). Notice that $2^{|C(u)|+2} \leq 2^{|C(v)|}$ as every gate in $C(u)$ appears in $C(v)$, too, but the gate v does not appear in $C(u)$ and the length of the entry v is at least two. With the same argument, we get $2^{|C(u_1)|+2} \leq 2^{|C(v)|}$ and $2^{|C(u_2)|+2} \leq 2^{|C(v)|}$.

(i) Let $\alpha(v) = \cup$.

(i.i) First, if $I(u_1) \subseteq \{0, \dots, 2^{|C(u_1)|}\}$ and $I(u_2) \subseteq \{0, \dots, 2^{|C(u_2)|}\}$, then

$$I(v) = I(u_1) \cup I(u_2) \subseteq \{0, \dots, 2^{|C(u_1)|}\} \cup \{0, \dots, 2^{|C(u_2)|}\} \subseteq \{0, \dots, 2^{|C(v)|}\}$$

(i.ii) Secondly, we have the case that for at least one u_i, we have $\overline{I(u_i)} \subseteq \{0, \dots, 2^{|C_{u_i}|}\}$. Without loss of generality assume $i = 1$. Then

$$\begin{aligned}\overline{I(v)} &= \overline{I(u_1) \cup I(u_2)} = \overline{I(u_1)} \cap \overline{I(u_2)} \subseteq \overline{I(u_1)} \subseteq \{0, \dots, 2^{|C(u_1)|}\} \\ &\subseteq \{0, \dots, 2^{|C(v)|}\}.\end{aligned}$$

(ii) Let $\alpha = \cap$.

(ii.i) Consider the case, where for at least one u_i we have $I(u_1) \subseteq \{0, \dots, 2^{|C_{u_i}|}\}$, without loss of generality $i = 1$. Then

$$I(v) = I(u_1) \cap I(u_2) \subseteq I(u_1) \subseteq \{0, \dots, 2^{|C_{u_i}|}\} \subseteq \{0, \dots, 2^{|C(v)|}\}.$$

(ii.ii) Now consider the case $\overline{I(u_1)} \subseteq \{0, \dots, 2^{|C(u_1)|}\}$ and $\overline{I(u_2)} \subseteq \{0, \dots, 2^{|C(u_2)|}\}$, then

$$\begin{aligned}\overline{I(v)} &= \overline{I(u_1) \cap I(u_2)} = \overline{I(u_1)} \cup \overline{I(u_2)} \subseteq \{0, \dots, 2^{|C(u_1)|}\} \cup \{0, \dots, 2^{|C(u_1)|}\} \\ &\subseteq \{0, \dots, 2^{|C(v)|}\}.\end{aligned}$$

(iii) Let $\alpha(v) = \overline{\ }$, then either

$$\begin{aligned}&\overline{I(v)} = I(u_1) \subseteq \{0, \dots, 2^{|C_u|}\} \subseteq \{0, \dots, 2^{|C(v)|}\} \text{ or} \\ &I(v) = \overline{I(u_1)} \subseteq \{0, \dots, 2^{|C_u|}\} \subseteq \{0, \dots, 2^{|C(v)|}\}.\end{aligned}$$

(iv) Let $\alpha(v) = +$. Assume $I(u_1) \neq \emptyset \neq I(u_2)$, otherwise we have

$$I(v) = \emptyset \subseteq \{0, \ldots, 2^{|C(v)|}\}.$$

(iv.i) First consider $I(u_1) \subseteq \{0, \ldots, 2^{|C(u_1)|}\}$ and $I(u_2) \subseteq \{0, \ldots, 2^{|C(u_2)|}\}$. Without loss of generalit let $|C(u_1)| \geq |C(u_2)|$ then

$$\begin{aligned} I(v) = I(u_1) + I(u_2) &\subseteq \{0, \ldots, 2^{|C(u_1)|}\} + \{0, \ldots, 2^{|C(u_2)|}\} \\ &= \{0, \ldots, 2^{|C(u_1)|} + 2^{|C(u_2)|}\} \subseteq \{0, \ldots, 2 \cdot 2^{|C(u_1)|}\} \\ &= \{0, \ldots, 2^{|C(u_1)|+1}\} \subseteq \{0, \ldots, 2^{|C(v)|}\}. \end{aligned}$$

(iv.ii) Consider the case where for at least one u_i we have $\overline{I(u_i)} \subseteq \{0, \ldots, 2^{|C_{u_i}|}\}$, without loss of generality $i = 1$. Notice that if $I(u_2) \subseteq \{0, \ldots, 2^{|C(u_2)|}\}$ or $\overline{I(u_2)} \subseteq \{0, \ldots, 2^{|C(u_2)|}\}$ in both cases there is an $a \in I(u_2)$ satisfying $a \leq 2^{|C(u_2)|} + 1$. Now consider for $x > 2^{|C(v)|}$ the sum $x = (x - a) + a$. If we can show $x - a > 2^{|C(u_1)|}$, then we get $x - a \in I(u_1)$ and $x \in I(u_1) + I(u_2) = I(v)$ for all $x > 2^{|C(v)|}$, so subsequently $\overline{I(v)} \subseteq \{0, \ldots, 2^{|C(v)|}\}$. Without loss of generality let $|C(u_1)| \geq |C(u_2)|$, then we have

$$2^{|C(u_1)|} + (2^{|C(u_2)|} + 1) \leq 2 \cdot 2^{|C(u_1)|} + 1 = 2^{|C(u_1)|+1} + 1 \leq 2^{|C(v)|}.$$

We use that to calculate

$$\begin{aligned} x - a &> 2^{|C(v)|} - a \geq 2^{|C(u_1)|} + (2^{|C(u_2)|} + 1) - a \geq 2^{|C(u_1)|} + a - a \\ &= 2^{|C(u_1)|} \end{aligned}$$

(v) Let $\alpha(v) = \min$. If $I(u) = \emptyset$, then $I(v) = \emptyset \subseteq \{0, \ldots, 2^{|C(v)|}\}$, so let $I(u) \neq \emptyset$. As already seen above, there exists an $a \in I(u)$ with $a \leq 2^{|C(u)|} + 1$, so

$$I(v) \subseteq \{0, \ldots, 2^{|C(u)|} + 1\} \subseteq \{0, \ldots, 2^{|C(v)|}\}.$$

(vi) Let $\alpha(v) = \max$.

(vi.i) If $I(u) \subseteq \{0, \ldots, 2^{|C(u)|}\}$, then $\max I(u) \in \{0, \ldots, 2^{|C(u)|}\}$, so

$$I(v) = \{\max I(u)\} \subseteq \{0, \ldots, 2^{|C(u)|}\} \subseteq \{0, \ldots, 2^{|C(v)|}\}$$

(vi.ii) If $\overline{I(u)} \subseteq \{0, \ldots, 2^{|C(u)|}\}$, then the set $I(u)$ is infinite, so

$$I(v) = \emptyset \subseteq \{0, \ldots, 2^{|C(v)|}\}.$$

■

Theorem 6.3
Let $\mathcal{O} \subseteq \{\cup, \cap, {}^{-}, +\}$ *and* $\mathcal{M} \coloneqq \{\min, \max\}$. *Then*

$$\mathrm{MC}(\mathcal{O} \cup \mathcal{M}) \in \mathrm{PSPACE}.$$

Proof. Let $C \coloneqq (V, E, v_C, \alpha)$ be a $(\cup, \cap, \overline{\ }, +, \min, \max)$-circuit and $b \in N$. Further let $oracle(C, b)$ be a PSPACE-algorithm for solving $\mathrm{MC}(\cup, \cap, \overline{\ }, +)$, which is possible due to [MW07]. Consider the following algorithm

Algorithm 6.1 Solving $\mathrm{MC}(\cap, \cup, \overline{\ }, +, \min, \max)$

Input: $(\cap, \cup, \overline{\ }, +, \min, \max)$-circuit $C \coloneqq (V, E, v_C, \alpha)$ and $b \in \mathbb{N}$

```
1: Add two new gates v_0, α(v_0) := 0 and v_1, α(v_1) := 1 at the beginning of C.
2: for every v ∈ V do
3:     if α(v) = min then
4:         Find the predecessor u of v and delete the edge (u, v) from E.
5:         for 0 ≤ i ≤ 2^|C| do
6:             if oracle(C_u, i) returns true then
7:                 Overwrite the label α(v) := i.
8:                 Continue the outer for-loop with the next v ∈ V.
9:         Set v as successor of v_0 and v_1 and overwrite the label α(v) = ∩.
10:    if α(v) = max then
11:        Find the predecessor u of v and delete the edge (u, v) from E.
12:        if oracle(C_u, 2^|C| + 1) returns true then
13:            Set v as successor of v_0 and v_1 and overwrite the label α(v) = ∩.
14:            Continue the outer for-loop with the next v ∈ V.
15:        for 2^|C| ≥ i ≥ 0 do
16:            if oracle(C_u, i) returns true then
17:                Overwrite the label α(v) := i.
18:                Continue the outer for-loop with the next v ∈ V.
19:        Set v as successor of v_0 and v_1 and overwrite the label α(v) = ∩.
20: Return oracle(C, b).
```

With Lemma 6.2 in mind, we see that this algorithm substitutes all min-, max-gate with the proper value. Let us analyze the needed space. First we add two new gates to C, whose number of possible successors is bounded by $|V|$, so this can be done in polynomial space. Next the algorithm generates numbers up to $2^{|C|}+1$, whose length is bounded by $|C|+1$. Then the algorithm computes *oracle*, where the size of its input is bounded polynomially, resulting in polynomially bounded space. ■

Corollary 6.4
Let $\mathcal{O} \subseteq \{\cap, \cup, \overline{\ }, +\}$ *and* $\mathcal{M} \coloneqq \{\min, \max\}$. *Then*

$$\mathrm{MF}(\mathcal{O} \cup \mathcal{M}) \in \mathrm{PSPACE}$$

Proof. Let $F \coloneqq (V, E, v_F, \alpha)$ be a $(\mathcal{O} \cup \mathcal{M})$-formula and $b \in \mathbb{N}$. Applying Algorithm 6.1 solves the question, if $(F, b) \in \mathrm{MF}(\mathcal{O} \cup \mathcal{M})$. ■

6.2 $(\cup, \cap, +, \min, \max)$-formulas

Previously in section 5.3 we found Δ_2^{P} as an upper bound for $\mathrm{MF}(\cup, \cap, +, \times, \min, \max)$. In this section, we will see that Δ_2^{P} is also a lower bound for $\mathrm{MF}(\cup, \cap, +, \min, \max)$ regarding many-one polynomial time reductions.

We will show the Δ_2^{P}-hardness of $\mathrm{MF}(\cup, \cap, +, \min, \max)$ by showing

$$\text{MAX SOS}_{\text{odd}} \leq_m^{\mathrm{P}} \mathrm{MF}(\cup, \cap, +, \min, \max).$$

Here MAX SOS_{odd} is a variation of the problem MAX SOS, which is in turn the generalized version of the famous Sum-of-subsets problem (SOS) defined now

$$\text{SOS} = \{(a_1, \ldots, a_n, b) \mid n, a_1, \ldots, a_n, b \in \mathbb{N} \text{ and there exists } \alpha_i \in \{0,1\} \text{ such that } \sum_{i=1}^{n} \alpha_i \cdot a_i = b\}.$$

The Δ_2^{P}-completeness of MAX SOS_{odd} was shown by Wagner in [Wag87].

Definition 6.5. Define

$$\text{MAX SOS} \coloneqq \{(a_1, \ldots, a_n, b, c) \mid n, a_1, \ldots, a_n, b, c \in \mathbb{N} \text{ and } \max M_{\text{SOS}}(a_1, \ldots, a_n, b) \geq c\},$$

where $M_{\text{SOS}}(a_1, \ldots, a_n, b) \coloneqq \{\sum_{i=1}^{n} \alpha_i \cdot a_i \mid \alpha_1, \ldots, \alpha_n \in \{0,1\} \text{ and } \sum_{i=1}^{n} \alpha_i \cdot a_i \leq b\}$ denotes the set of all possible sums over $a_1, \ldots, a_n$ being less or equal than b. This means a list of numbers $(a_1, \ldots, a_n, b, c)$ is an element of MAX SOS if and only if the biggest possible sum over $a_1, \ldots, a_n$ being less or equal than b is also greater or equal than c. From that we define

$$\text{MAX SOS}_{\text{odd}} \coloneqq \{(a_1, \ldots, a_n, b, c) \mid (a_1, \ldots, a_n, b, c) \in \text{MAX SOS and } \max M_{\text{SOS}}(a_1, \ldots, a_n, b) \text{ is an odd number}\},$$

so a list of numbers $(a_1, \ldots, a_n, b, c)$ is in MAX SOS_{odd} if and only if the biggest possible sum over $a_1, \ldots, a_n$ being less or equal than b is greater or equal than c and additionally odd.

So let $(a_1, \ldots, a_n, b, c)$ be a list of numbers and a possible instance of MAX SOS_{odd}. The goal now is to find a function $f \in \mathrm{FP}$ such that

$$(a_1, \ldots, a_n, b, c) \in \text{MAX SOS}_{\text{odd}} \Leftrightarrow f(a_1, \ldots, a_n, b, c) \in \mathrm{MF}(\cup, \cap, +, \min, \max),$$

where $f(a_1, \ldots, a_n, b, c) = (F, b)$ with a formula F and a number b. We describe this formula F by using the following subformulas as building blocks. We will explain how to generate these subformulas (especially in polynomial time) later on. Let

- $SUM_{a_1 \ldots, a_n} \coloneqq \{\sum_{i=1}^{n} \alpha_i \cdot a_i \mid \alpha_i \in \{0,1\}\}$ be the sum of all possible sums over the list of numbers $a_1, \ldots, a_n$.

- $INT_{c,b} \coloneqq [c, b]$ be the natural, closed interval from c to b.
- $ODDINT_{c,b} \coloneqq [c, b] \cap 2\mathbb{N} + 1$ be the set of all odd numbers between c and b.

From that define F by

$$F \coloneqq (\max(SUM_{a_1,\ldots,a_n} \cap INT_{c,b}) \cap ODDINT_{c,b}) + INT_{0,b-c}.$$

Then

$$(a_1, \ldots, a_n, b, c) \in \text{MAX SOS}_{\text{odd}} \Leftrightarrow b \in I(F).$$

Before proofing this claim, we first take a look at how to generate these subformulas in polynomial time. The question is: How do we generate formulas in polynomial time that compute the desired sets? For example taking the naive approach for the interval [c,b] by taking the union of all numbers $INT_{c,b} = c \cup (c+1) \cup \cdots \cup b$ one needs $b - c$-many gates. Considering a sufficiently large b and a small c one can see, that these are exponentially many gates, which would violate the polynomial time restriction!

Lemma 6.6
Let $(a_1, \ldots, a_n) \in \mathbb{N}^n$ be a list of numbers Then the formula

$$SUM_{a_1,\ldots,a_n} \coloneqq \sum_{i=1}^{n} (0 \cup a_i)$$

can be generated in polynomial time and for its computed set, we get

$$I(SUM_{a_1,\ldots,a_n}) = \left\{ \sum_{i=1}^{n} \alpha_i \cdot a_i \mid \alpha_i \in \{0, 1\} \right\}$$

Proof. Let $x \in I(SUM_{a_1,\ldots,a_n})$. Then

$$x = \sum_{i=1}^{n} x_i, \text{ where } x_i \in \{0, a_i\}.$$

We can rewrite this sum as

$$x = \sum_{i=1}^{n} \alpha_i \cdot a_i, \text{ with } \alpha_i \in \{0, 1\},$$

so $I(SUM_{a_1,\ldots,a_n}) \subseteq \{\sum_{i=1}^{n} \alpha_i \cdot a_i \mid \alpha_i \in \{0, 1\}\}$. One can show the other direction analogously, so the equality

$$I(SUM_{a_1,\ldots,a_n}) = \left\{ \sum_{i=1}^{n} \alpha_i \cdot a_i \mid \alpha_i \in \{0, 1\} \right\}$$

is true. The construction is realized by the following algorithm, its runtime is bounded by $O(n)$. See Figure 6.1 for a visualization of this algorithm.

Algorithm 6.2 Algorithm for generating $SUM_{a_1,\ldots,a_n}$

Input: A list of numbers $(a_1, \ldots, a_n) \in \mathbb{N}^n$.

if $n = 0$, i.e. the input is an empty list, **then**
    Write the gate $v_1, 0$ onto the output and set v_1 as the output gate.
    Stop here the algorithm and return this formula consisting of a single gate.
for $1 \leq i \leq n$ **do**
    Write the gate v_i, a_i, c_i onto the output.
    Write the gate $z_i, 0, c_i$ onto the output.
if $n = 1$ **then**
    Write the gate $c_1, \cup$ onto the output and set it as the output gate.
    At this point the formula is complete, so stop here.
Write the gate $c_1, \cup, p_2$ onto the output.
for every $2 \leq i \leq n$ **do**
    Write the gate $c_i, \cup, p_i$ onto the output.
for every $2 \leq i \leq n-1$ **do**
    Write the gate $p_i, +, p_{i+1}$ onto the output.
Write the gate $p_n, +$ onto the output and set it as the output gate.

■

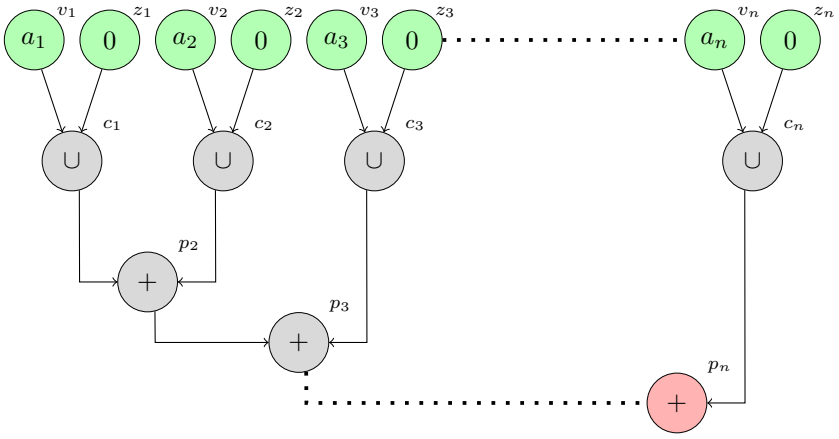


Figure 6.1: Visualization of Algorithm 6.2

Before continuing with the next building blocks, consider the following Lemma.

Lemma 6.7
Let F and G be two formulas.

(i) If $I(F) = [0, n]$ and $I(G) = [0, m]$ for some $n, m \in \mathbb{N}$. Then

$$I(F + G) = [0, n + m]$$

(ii) If $I(F) = [0,n] \cap 2\mathbb{N}$ and $I(G) = [0,m] \cap 2\mathbb{N}$ for some $n, m \in \mathbb{N}$. Then

$$I(F+G) = [0, n+m] \cap 2\mathbb{N}$$

Proof. (i) Let $x \in I(F+G)$, i.e. $x = f + g$ for some $f \leq n$ and $g \leq m$. Then $x \leq n+m$ and $x \in [0, n+m]$, so $I(F+G) \subseteq [0, n+m]$.

Now let $x \in [0, n+m]$. Without loss of generality let $n \leq m$. Consider the following cases.

- If $x \leq n$, then $x = x + 0$ with $x \in [0,n] = I(F)$ and $0 \in [0,m] = I(G)$.
- If $n < x \leq n+m$, then $x = n + (x-n)$ with $n \in [0,n] = I(F)$ and $x - n \in [0,m] = I(G)$.

In both cases we get $x \in I(F+G)$, so $[0, n+m] \subseteq I(F+G)$.

(ii) Let $x \in I(F+G)$, i.e. $x = f + g$ for some $f \leq n$ and $g \leq m$ both being even. Then $x \leq n+m$, so $x \in [0, n+m]$ and x is even as it is the sum of two even numbers. So $I(F+G) \subseteq [0, n+m] \cap 2\mathbb{N}$.

Now let $x \in [0, n+m] \cap 2\mathbb{N}$. Without loss of generality let $n \leq m$. Consider the following cases.

- If $x \leq n$, then $x = x + 0$ with $x \in [0,n] \cap 2\mathbb{N} = I(F)$ and $0 \in [0,m] \cap 2\mathbb{N} = I(G)$.
- If $n < x \leq n+m$, distinguish the following cases. If n is even, then $x = n + (x-n)$ with $n \in [0,n] \cap 2\mathbb{N} = I(F)$. Further $x - n$ is even as x and n are even, so $x - n \in [0,m] = I(G)$.

  If n is odd, then $[0,n] \cap 2\mathbb{N} = [0, n-1] \cap 2\mathbb{N}$ and $x \leq (n-1) + m$. Consider $x = (n-1) + (x - n + 1)$, then $n - 1 \in [0,n] \cap 2\mathbb{N} = I(F)$. As x is even and n is odd, $x - n + 1$ is even, so $x - n + 1 \in [0,m] = I(G)$.

In all cases, we get $x \in I(F+G)$, so $[0, n+m] \cap 2\mathbb{N} \subseteq I(F+G)$.

■

Lemma 6.8
Let $(b,c) \in \mathbb{N}^2$ satisfying $c \leq b$. Further let $b - c = \sum_{j=0}^{k} \beta_j \cdot 2^i$ with $\beta_j \in \{0,1\}$ be the binary representation of $b - c$ and let $J \coloneqq \{j \mid \beta_j = 1\}$ be the set of the indices equal 1 of said representation. Analogous let $\hat{J} \coloneqq \{j \mid \hat{\beta}_j = 1\}$ be the indices equal 1 of the binary representation of $c - b - 1$. Then the following formulas can be generated in polynomial time and the stated equalities are true.

(i) Define

$$INT_{c,b} \coloneqq \sum_{j \in J} \left(2^j \cup \sum_{i=0}^{j-1} (0 \cup 2^i) \right) + c.$$

Here for $j = 0$ the empty sum is defined as $\sum_{i=0}^{0-1}(0 \cup 2^i) = 0$. For the computed set we get

$$I(INT_{c,b}) = [c, b].$$

(ii) If c is odd, define

$$ODDINT_{c,b} \coloneqq \sum_{\substack{j\in J\\ j\neq 0}} \left(2^j \cup \sum_{i=1}^{j-1}(0 \cup 2^i)\right) + c.$$

If c is even, define

$$ODDINT_{c,b} \coloneqq \sum_{\substack{j\in \hat{J}\\ j\neq 0}} \left(2^j \cup \sum_{i=1}^{j-1}(0 \cup 2^i)\right) + c.$$

In both cases we get

$$I(ODDINT_{c,b}) = [c,b] \cap 2\mathbb{N} + 1.$$

Proof. (i) Considering the inner sum of the left side for some fixed $j \in J$, with Lemma 6.6 we get

$$2^j \cup \sum_{i=0}^{j-1}(0 \cup 2^i) = 2^j \cup \left\{\sum_{i=0}^{j-1} \alpha_i \cdot 2^i \mid \alpha_i \in \{0,1\}\right\} = 2^j \cup \left[0, 2^j - 1\right] = \left[0, 2^j\right].$$

Now we compute

$$\begin{aligned} &\sum_{j\in J}\left(2^j \cup \sum_{i=0}^{j-1}(0 \cup 2^i)\right) + c = \left(\sum_{j\in J}[0,2^j]\right) + c = [c,b] \\ \Leftrightarrow &\sum_{j\in J}[0,2^j] = [0, b-c], \end{aligned}$$

where the last line is indeed true as $\sum_{j\in J} 2^j = b - c$ and applying Lemma 6.7 iteratively.

(ii) If c is odd, we have

$$\begin{aligned} &\sum_{\substack{j\in J\\ j\neq 0}}\left(2^j \cup \sum_{i=1}^{j-1}(0 \cup 2^i)\right) + c = [c,b] \cap 2\mathbb{N} + 1 \\ \Leftrightarrow &\sum_{\substack{j\in J\\ j\neq 0}}\left(2^j \cup \sum_{i=1}^{j-1}(0 \cup 2^i)\right) = [0, b-c] \cap 2\mathbb{N}. \end{aligned}$$

For some fixed $j \in J$, we get from Lemma 6.7

$$2^j \cup \sum_{i=1}^{j-1}(0 \cup 2^i) = [0, 2^j] \cap 2\mathbb{N}. \tag{6.1}$$

This yields for the whole sum

$$\begin{aligned}\sum_{\substack{j\in J\\ j\neq 0}}\left(2^j \cup \sum_{i=1}^{j-1}(0\cup 2^i)\right) &= \sum_{\substack{j\in J\\ j\neq 0}}\left([0,2^j]\cap 2\mathbb{N}\right) \overset{1}{=} \sum_{j\in J}\left([0,2^j]\cap 2\mathbb{N}\right)\\ &= [0,b-c]\cap 2\mathbb{N}.\end{aligned} \tag{6.2}$$

If c is even, then $c-1$ is odd, so applying the argumentation in (6.1) and (6.2) immediately yields

$$\sum_{\substack{j\in \hat{J}\\ j\neq 0}}\left(2^j \cup \sum_{i=1}^{j-1}(0\cup 2^i)\right) = [0,b-c+1]\cap 2\mathbb{N}.$$

We conclude

$$\begin{aligned}&\sum_{\substack{j\in \hat{J}\\ j\neq 0}}\left(2^j \cup \sum_{i=1}^{j-1}(0\cup 2^i)\right) + c = [c,b]\cap(2\mathbb{N}+1)\\ \Leftrightarrow\ &([0,b-c]\cap 2\mathbb{N}) + c = [c,b]\cap(2\mathbb{N}+1)\\ \Leftrightarrow\ &([0,b-c]\cap 2\mathbb{N}) = ([0,b-c]\cap 2\mathbb{N})\end{aligned}$$

Before implementing the generation of these formulas, consider first the following polynomial time algorithm for efficiently creating a list representing the binary representation of a given number.

Algorithm 6.3 Algorithm for generating the binary representation of n.

Input: A number $n \in \mathbb{N}$.

1: Set $e \coloneqq 0$, $p \coloneqq 1$, $x \coloneqq n$ and initialize an empty list J.
2: **while** $2 \cdot p \leq x$ **do**
3: Increment e by 1 and double the value of p.
4: **for** $0 \leq i \leq e$ **do**
5: **if** $x > p$ **then**
6: Set $x \coloneqq x - p$ and store $J[e-i] \coloneqq 1$.
7: **else**
8: Store $J[e-i] \coloneqq 0$.
9: Double the value of x.
10: Return the list J.

This is indeed a polynomial time algorithm, as the number of steps performed in the while-loop (as well as the for-loop by construction) is bounded by $O(|p|)$, which is in turn bounded by $O(|n|)$.

[1] As for $j = 0$ the inner term would be $[0,1]\cap 2\mathbb{N} = \{0\}$.

Now we state an algorithm for generating both $INT_{c,b}$ and $ODDINT_{c,b}$ as these algorithms are very similar. Next to a pair of numbers (b, c) the algorithm will have a variable $k \in \{0, 1\}$ as its input, which signals, if $INT_{c,b}$ or $ODDINT_{c,b}$ shall be generated. (If $k = 0$, then $INT_{c,b}$ will be generated, if $k = 1$, then $ODDINT_{c,b}$ will be generated.) Check Figure 6.2 for a visualization of this algorithm.

Algorithm 6.4 Algorithm for generating $INT_{c,b}$ and $ODDINT_{c,b}$.

Input: A pair of numbers $(c, b) \in \mathbb{N}^2$ satisfying $c < b$ and $k \in \{0, 1\}$.

1: **if** $k = 0$ or $b - c$ is odd **then**
2: Apply Algorithm 6.3 on the input $b - c$ and store its result in the list J.
3: **else**
4: Apply Algorithm 6.3 on the input $b - c - 1$ and store its result in the list J.
5: Let l_J be the length of J and set $l \coloneqq l_J - 1$.
6: **for** $k \leq j \leq l$ **do**
7: **if** $J[j] = 1$ **then**
8: Compute the list $S \coloneqq (2^k, 2^{1+k}, \ldots, 2^{j-1})$.
9: Let F_j be the output of Algorithm 6.2 applied on the list S.
10: Write F_j onto the output and set c_j as the successor of F's output gate.
11: Write the gate $u_j, 2^j, c_j$ onto the output.
12: Write the gate $c_j, \cup, p_j$ onto the output.
13: Let $k \leq j_1$ be the first entry of J, such that $J[j_1] = 1$.
14: Write the gate v, c, p_{j_1} onto the output
15: **for** $j_1 \leq j < l$ **do**
16: **if** $J[j] = 1$ **then**
17: Find the next greater j' such that $J[j'] = 1$.
18: Write the gate $p_j, +, p_{j'}$ onto the output.
19: Write the gate $p_l, +$ and mark this gate as the output gate.[2]

First Algorithm 6.3 is performed on the input $b-c$ or $b-c-1$, whose length is bounded by the length of the input (c, b). We already stated above that the runtime of this algorithm is polynomially bounded.

Next the number of steps performed in the two for-loops (starting in line 6 resp. line 15) is essentially equal to the length l of the list J. This length is bounded by the length of $b - c$, so we can conclude the number of iterations performed by the two for loops is polynomially bounded by the length of the input.

Concerning the runtime within these iterations, the critical parts are lines 8 and 9. The list S computed in line 8 has at most l-many elements of size at most 2^{l-1}, whose length is $l-1$. So the total length is bounded by l^2+l, which is polynomially bounded by the length of the input. Then in line 9 we perform on this polynomially bounded list Algorithm 6.2, which has according to Lemma 6.6 a polynomially bounded runtime. In total this yields polynomially bounded runtime for one of our iterations.

[2] As by construction, we did not store leading zeros in J, so we know that $J[l] = 1$.

In recap, we have to perform a polynomially bounded number of iterations, where each iteration has polynomially runtime. This yields the runtime of Algorithm 6.4 is polynomially bounded. ■

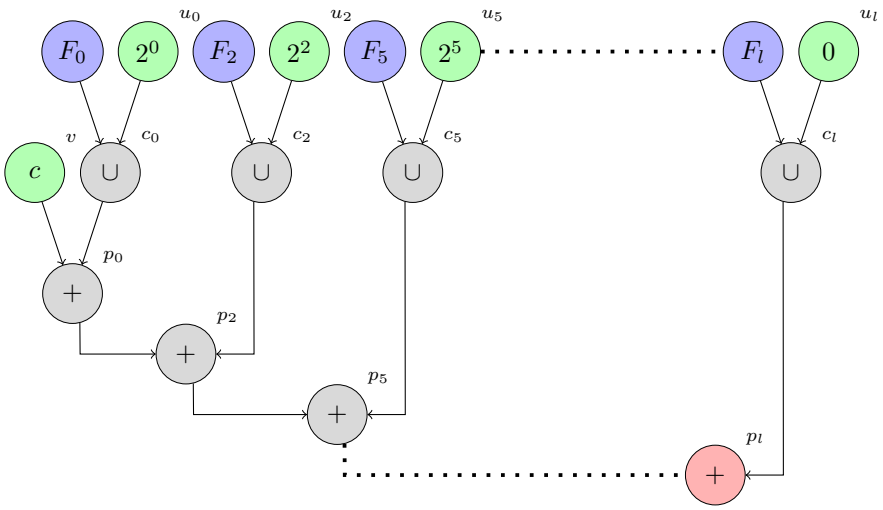


Figure 6.2: Visualization of Algorithm 6.4. In this example, assume $INT_{c,b}$ is being computed and the list J start with the sequence $(1, 0, 1, 0, 0, 1)$.

Remark. Consider Algorithm 6.4. Technically, the iterative calls of Algorithm 6.2 create the same gate names over and over again, leading to not unique gate names in our final formula. This issue can be circumvented easily by modifying Algorithm 6.2 in the following way. Create a new input number n and attach this number to every gate name generated by the algorithm. In Algorithm 6.4 add the variable j to the input when calling Algorithm 6.2. Then the name of every gate generated by the subroutine Algorithm 6.2 contains the information, in which iterative call it was created - leading to unique names again.

A similar issue will arise in the upcoming algorithm, too. However, we can resolve this issue in the same way.

Theorem 6.9
$\mathrm{MF}(\cup, \cap, +, \min, \max)$ *is* Δ_2^{P}*-hard regarding polynomial time many-one reductions.*

Proof. As already stated, we will proof this statement by showing

$$\text{MAX SOS}_{\text{odd}} \leq_m^{\mathrm{P}} \mathrm{MF}(\cup, \cap, +, \min, \max).$$

We are searching for a function $f \in \mathrm{FP}$ such that the equivalence

$$(a_1, \ldots, a_n, b, c) \in \text{MAX SOS}_{\text{odd}} \Leftrightarrow f(a_1, \ldots, a_n, b, c) \in \mathrm{MF}(\cup, \cap, +, \min, \max)$$

is true. Define f via

$$f(a_1, \ldots, a_n, b, c) \coloneqq \begin{cases} (F, b), & \text{if } b \geq c \\ (G, 1), & \text{otherwise} \end{cases}$$

with formulas

$$F \coloneqq (\max(SUM_{a_1,\ldots,a_n} \cap INT_{c,b}) \cap ODDINT_{c,b}) + INT_{0,b-c} \text{ and}$$
$$G \coloneqq 1 + 1.$$

Let $(a_1, \ldots, a_n, b, c) \in \mathbb{N}^{n+2}$. First consider the case $c > b$. Then there exists no number greater or equal than c and less or equal than b. Consequently $(a_1, \ldots, a_n, b, c)$ can not be an element of MAX SOS_{odd}. In this case, f returns the formula $G = 1 + 1$ and the number $1 \in \mathbb{N}$. We see that $1 \notin I(G)$, so $(G, 1)$ is not an element of $\text{MF}(\cup, \cap, +, \min, \max)$.

Now consider the case $b \geq c$, where f returns the formula F as well as the number $b \in \mathbb{N}$. How does the set $I(F)$ look like? Let us use our knowledge about the subformulas $SUM_{a_1,\ldots,a_n}, INT_{c,b}$ and $ODDINT_{c,b}$ established in Lemma 6.6 and Lemma 6.8.

$$\begin{aligned} I(F) &= (\max(SUM_{a_1,\ldots,a_n} \cap INT_{c,b}) \cap ODDINT_{c,b}) + INT_{0,b-c} \\ &= \left(\max\left(\left\{ \sum_{i=1}^{n} \alpha_i \cdot a_i \mid \alpha_i \in \{0,1\} \right\} \cap [c,b] \right) \cap ([c,b] \cap (2\mathbb{N}+1)) \right) \\ &\quad + [0, b-c] \end{aligned} \tag{6.3}$$

First assume that $(a_1, \ldots, a_n, b, c) \in$ MAX SOS_{odd}. Then the greatest value of the set $\{\sum_{i=1}^{n} \alpha_i \cdot a_i \mid \alpha_i \in \{0,1\}\} \cap [c,b]$ exists, so

$$I(F) = (x \cap ([c,b] \cap 2\mathbb{N}+1)) + [0, b-c].$$

Additionally, we know x is odd (otherwise $(a_1, \ldots, a_n, b, c) \notin$ MAX SOS_{odd}), so

$$I(F) = x + [0, b-c] = [x, b-c+x].$$

We know $c \leq x \leq b$, so the left limit of the interval is never greater than b, while the right limit of the interval is never smaller than $b - c + c = b$. We conclude $b \in I(F)$ and $f(a_1, \ldots, a_n, b, c) = (F, b) \in \text{MF}(\cup, \cap, +, \min, \max)$.

Now assume that $(a_1, \ldots, a_n, b, c) \notin$ MAX SOS_{odd} and consider (6.3) once again. This means either that the set $\{\sum_{i=1}^{n} \alpha_i \cdot a_i \mid \alpha_i \in \{0,1\}\} \cap [c,b]$ is empty (i) or the greatest value in this set is even (ii). In case (ii) call this value again x.

(i) Then we have

$$I(F) = (\emptyset \cap ([c,b] \cap (2\mathbb{N}+1))) + [0, b-c] = \emptyset + [0, b-c] = \emptyset,$$

(ii) Then we have

$$I(F) = (x \cap ([c,b] \cap (2\mathbb{N}+1))) + [0, b-c] = \emptyset + [0, b-c] = \emptyset,$$

In both cases, we get $b \notin I(F)$, so $f(a_1, \ldots, a_n, b, c) = (F, b) \notin \mathrm{MF}(\cup, \cap, +, \min, \max)$. We conclude that f is indeed a correct reduction. Further, f is realized by the following algoritm

Algorithm 6.5 Reducing MAX $\mathrm{SOS}_{\mathrm{odd}}$ to $\mathrm{MF}(\cup, \cap, +, \min, \max)$.

Input: A list of numbers $(a_1, \ldots, a_n, b, c) \in \mathbb{N}^{n+2}$.

1: **if** $b < c$ **then**
2: Generate the formula $F \coloneqq 1 + 1$ and return it along the number 1.
3: Let F_1 be the output of Algorithm 6.2 applied on the list $(a_1, \ldots, a_n)$.
4: Write F_1 onto the output and set c_1 as the successor of its output gate.
5: Let F_2 be the output of Algorithm 6.4 applied on (c, b) and $k = 0$.
6: Write F_2 onto the output and set c_1 as the successor of its output gate.
7: Write the gate $c_1, \cap, m$ onto the output.
8: Write the gate $m, \max, c_2$ onto the output.
9: Let F_3 be the output of Algorithm 6.4 applied on (c, b) and $k = 1$.
10: Write F_3 onto the output and set c_2 as the successor of its output gate.
11: Write the gate $c_2, \cap, p$ onto the output.
12: Let F_4 be the output of Algorithm 6.4 applied on $(0, b - c)$ and $k = 0$.
13: Write F_4 onto the output and set p as the successor of its output gate.
14: Write the gate $p, +$ onto the output and set it as the output gate.

This algorithm's runtime is polynomially bounded in its input size, as it applies polynomial runtime Algorithms (see Lemma 6.6 and Lemma 6.8) on inputs, whose sizes are bounded by the size of its own input. This shows $f \in \mathrm{FP}$, which concludes this proof. ■

6.3 The difficulty of $\mathrm{MC}(\cup, \cap, \times, \min, \max)$ and $\mathrm{MC}(\cup, \cap, \overline{\ }, \times, \min, \max)$

Let $\mathcal{O} \subseteq \{\cup, \cap, \times\}$ and $\mathcal{M} \coloneqq \{\min, \max\}$. Trying to find complexity classes for the problems $\mathrm{MC}(\mathcal{O} \cup \mathcal{M})$, $\mathrm{MC}(\{\overline{\ }\} \cup \mathcal{O} \cup \mathcal{M})$ as well as their respective formulas counterparts either as lower or upper bounds turned out to be rather difficult.

Remember Theorem 6.3 for solving $\mathrm{MC}(\cup, \cap, +, \min, \max)$, the idea there was to eliminate a min, max-gate v of a given circuit C by testing for $i \in \mathbb{N}$ up to (resp. downwards in case of a max-gate) a certain threshold, if $i \in I(v)$. This was possible in polynomial space because the length of this threshold was polynomially bounded. If we have $\times$ instead of $+$ however, one would have to consider all values up to threshold, whose length is only exponentially bounded. Using this idea leads to EXPSPACE as an upper bound. However with Theorem 5.17 we have already found a better upper bound in Δ_2^{EXP}.

McKenzie & Wagner showed $\mathrm{MC}(\cup, \cap, \times) \leq_m^{\mathrm{P}} \mathrm{MC}(\cup, \cap, +)$ by the following idea. Let $C \coloneqq (V, E, v_C, \alpha)$ be a $(\cup, \cap, \times)$-circuit. Decompose all numbers into a product of coprime numbers, resulting in a vector of exponents for all values. Instead of

multiplying two numbers, one can now add the elements of these vectors pairwise. Adding min-, max as possible gate labels this idea may fail, as it is unclear how to determine which of two decomposition describes the greater number in an effective way.

7 Conclusion

Let $\mathcal{M} \coloneqq \{\min, \max\}$.Then the results of this thesis are listed in the following table.

$\mathcal{O}$	MC($\mathcal{O} \cup \mathcal{M}$) lower bound		MC($\mathcal{O} \cup \mathcal{M}$) upper bound		MF($\mathcal{O} \cup \mathcal{M}$) lower bound		MF($\mathcal{O} \cup \mathcal{M}$) upper bound	
$\cup, \cap, {}^{-}, +, \times$	NEXP		?		PSPACE		?	
$\cup, \cap, +, \times$	NEXP		Δ_2^{EXP}	5.17	Δ_2^{P}	6.9	Δ_2^{P}	5.13
$\cup, +, \times$	PSPACE		PSPACE	5.7	NP		NP	5.8
$\cap, +, \times$	P		co-R	5.1	L		L	5.2
$+, \times$	P		P	5.1	L		L	5.2
$\cup, \cap, {}^{-}, +$	PSPACE		PSPACE	6.3	PSPACE		PSPACE	6.4
$\cup, \cap, +$	PSPACE		PSPACE	6.3	Δ_2^{P}	6.9	Δ_2^{P}	5.13
$\cup, +$	NP		NP	5.7	NP		NP	5.8
$\cap, +$	$\mathrm{C}_{=}\mathrm{L}$		$\mathrm{C}_{=}\mathrm{L}$	5.1	L		L	5.2
$+$	$\mathrm{C}_{=}\mathrm{L}$		$\mathrm{C}_{=}\mathrm{L}$	5.1	L		L	5.2
$\cup, \cap, {}^{-}, \times$	PSPACE		?		PSPACE		?	
$\cup, \cap, \times$	PSPACE		Δ_2^{EXP}	5.17	NP		Δ_2^{P}	5.13
$\cup, \times$	NP		NP	5.7	NP		NP	5.8
$\cap, \times$	$\mathrm{C}_{=}\mathrm{L}$		P	5.1	L		L	5.2
$\times$	NL		NL	5.1	L		L	5.2
$\cup, \cap, {}^{-}$	P		P	4.2	L		P	4.2
$\cup, \cap$	P		P	4.2	L		P	4.2
$\cup$	P	4.7	P	4.2	L		P	4.2
$\cap$	NL		NL	4.4	L		L	4.5
$\emptyset$	L		L	4.8	L		L	4.8

Here the lower bounds without citation are taken from Table 3.1, the studies of McKenzie and Wagner in [MW07]. Note that while MC($\cup, \min, \max$) is P-hard, we have seen in Theorem 4.6 that MC($\cup, \min$) $\in$ NL and MC($\cup, \max$) $\in$ NL.

The following questions could not be answered in this thesis and remain open.

(i) Is MF($\{\cup, \cap, {}^{-}\} \cup \mathcal{M}$) $\in$ L?

(ii) Is MC($\{\cup, \cap, {}^{-}, \times\} \cup \mathcal{M}$) $\in$ PSPACE and/or MF($\{\cup, \cap, {}^{-}, \times\} \cup \mathcal{M}$) $\in$ PSPACE? Is MC($\{\cup, \cap, \times\} \cup \mathcal{M}$) $\in$ PSPACE? Is MF($\{\cup, \cap, \times\} \cup \mathcal{M}$) Δ_2^{P}-hard?

(iii) Is MC($\{\cup, \cap, +, \times\} \cup \mathcal{M}$) Δ_2^{P}-hard?

References

[SM73] L. J. Stockmeyer, A. R. Meyer, Word problems requiring exponential time(preliminary report). Proceedings of the fifth annual ACM symposium on Theory of computing - STOC '73 (1973), doi:10.1145/800125.804029

[Gol77] L. M. Goldschlager, The monotone and planar circuit value problems are log space complete for p. ACM SIGACT News. 9, 25–29 (1977), doi:10.1145/1008354.1008356. .

[Wag87] K. W. Wagner, More complicated questions about maxima and minima, and some closures of np. Theoretical Computer Science. 51, 53–80 (1987), doi:10.1016/0304-3975(87)90049-1.

[MW07] P. McKenzie, K. W. Wagner, The complexity of membership problems for circuits over sets of natural numbers. computational complexity. 16, 211–244 (2007), doi:10.1007/s00037-007-0229-6.

Erklärung

"Hiermit versichere ich, dass ich die vorliegende Arbeit in allen Teilen selbstständig angefertigt und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt habe. Sämtliche wörtlichen oder sinngemäßen Übernahmen und Zitate sind kenntlich gemacht und nachgewiesen."

. . . . . .

Ort, Datum, Unterschrift