Satisfiability of Algebraic Circuits over Sets of Natural Numbers

Christian Glaßer, Christian Reitwießner,
Stephen Travers, and Matthias Waldherr

Theoretische Informatik, Julius Maximilians Universität Würzburg, Germany.

Abstract. We investigate the complexity of *satisfiability problems* for $\{\cup, \cap, ^{-}, +, \times\}$-circuits computing sets of natural numbers. These problems are a natural generalization of membership problems for expressions and circuits studied by Stockmeyer and Meyer (1973) and McKenzie and Wagner (2003).
Our work shows that satisfiability problems capture a wide range of complexity classes like NL, P, NP, PSPACE, and beyond. We show that in several cases, satisfiability problems are harder than membership problems. In particular, we prove that testing satisfiability for $\{\cap, +, \times\}$-circuits already is undecidable. In contrast to this, the satisfiability for $\{\cup, +, \times\}$-circuits is decidable in PSPACE.

Classification: Computational Complexity; Combinatorial Integer Circuits; Algorithms

1 Introduction

In complexity theory, satisfiability questions play an important role in understanding the nature of computational problems. The satisfiability test for *Boolean formulas* is the question of whether there exists an assignment of truth values *true* and *false* to the variables such that the Boolean expression evaluates to true. This was the first natural problem proven to be NP-complete [Coo71] and it is still one of the most prominent NP-complete problems today. The latter also holds for the similar problem of testing satisfiability for *boolean circuits*, where boolean expressions are described in a more succinct way.

In this paper, we investigate satisfiability questions for a more general kind of circuits, namely algebraic circuits over sets of natural numbers. The notion of algebraic circuits has its origin in *Integer Expressions* introduced by Stockmeyer and Meyer [SM73] in 1973. An integer expression is an expression built up from single natural numbers by using set operations ($^{-}$, $\cup$, $\cap$) and algebraic operations ($+$, $\times$). Stockmeyer and Meyer investigated the complexity of *membership problems* for such expressions, i.e., given an expression E, how difficult is it to test whether a certain natural number is a member of the set described by E? Restricting the set of allowed operations results in membership problems of different complexities.

Wagner [Wag84] introduced *circuits over sets of natural numbers* in 1984. The latter describe integer expressions in a more succinct way. The input gates of such a circuit are labeled with natural numbers, the inner gates compute set operations ($^{-}$, $\cup$, $\cap$) and arithmetic operations ($+$,$\times$). Wagner [Wag84], Yang [Yan00], and McKenzie and Wagner [MW03] studied the complexity of membership problems for algebraic circuits over natural numbers: Here, for a given circuit C with given numbers assigned to the input gates, one has to decide whether a given number n belongs to the set described by C.

In this paper, we study generalizations of these membership problems, namely *satisfiability problems* for algebraic circuits over sets of natural numbers. In contrast to membership problems, here a circuit can contain *unassigned input gates.* The question is, given a circuit C with gate labels from $\mathcal{O}$, $\mathcal{O} \subseteq \{^-, \cup, \cap, +, \times\}$, and given a natural number n, does there exist an assignment of natural numbers to the variable input gates such that n is contained in the set described by the circuit? We denote this problem with $\mathrm{SC}(\mathcal{O})$.

As our circuits can still contain non-variable input gates with fixed inputs, it is immediate that a satisfiability problem always is a generalization of a membership problem. Hence, solving a satisfiability problem is at least as hard as solving a membership problem.

Notice that the domain of the input variables is unbounded, hence it is not a priori clear that our satisfiability problems are decidable. Nevertheless, we can characterize the complexity of many satisfiability problems precisely by proving them to be complete for (decidable) complexity classes. In other cases however, we can formally prove the satisfiability problem to be undecidable: We show that the problem of solving diophantine equations, which was proven to be undecidable by Matiyasevich [DPR61,Mat70], can be reduced to $\mathrm{SC}(\cap, +, \times)$, the problem of testing satisfiability for $\{\cap, +, \times\}$-circuits.

Interestingly, if we start with $\mathrm{SC}(\cap, +, \times)$ and drop one of the operations $\cap$, $+$, or $\times$, then in all three cases we arrive at an NP-complete problem, namely $\mathrm{SC}(+, \times)$, $\mathrm{SC}(\cap, +)$, or $\mathrm{SC}(\cap, \times)$. The latter is of particular interest, since in contrast to most other NP-complete problems, here the membership in NP is more difficult to show than the NP-hardness. For this end, we introduce a problem that addresses the solvability of certain *systems of monom equations.* The nontrivial fact that integer programming is contained in NP allows us to show that the solvability of systems of monom equations also belongs to NP. Finally, this can be used to establish $\mathrm{SC}(\cap, \times) \in \mathrm{NP}$.

Our main open question is whether $\mathrm{SC}(^-, \cup, \cap, \times)$, the satisfiability problem for $\{^-, \cup, \cap, \times\}$-circuits, is decidable. A further open question is to find a better lower bound for the satisfiability problem for $\{\times\}$-circuits. We prove this problem to be in $\mathrm{UP} \cap \mathrm{coUP}$.

A summary of our results (Table 1) and a discussion of open problems can be found in the conclusions section.

2 Preliminaries

We fix the alphabet $\Sigma = \{0, 1\}$. Σ^* is the set of words, and $|w|$ is the length of a word $w \in \Sigma^*$. $\mathbb{N}$ denotes the set of the natural numbers, $\mathbb{N}^+$ denotes the set of positive integers. We denote with L, NL, P, NP, coNP, and PSPACE the standard complexity classes whose definitions can be found in textbooks on computational complexity [Pap94].

We extend the arithmetical operations $+$ and $\cdot$ to subsets of $\mathbb{N}$: Let $M, N \subseteq \mathbb{N}$. We define the sum of M and N as $M + N \overset{\mathit{df}}{=} \{m + n : m \in M \text{ and } n \in N\}$. We define the product of M and N as $M \times N \overset{\mathit{df}}{=} \{m \cdot n : m \in M \text{ and } n \in N\}$. Unless otherwise stated, the domain of a variable is $\mathbb{N}$.

For a complexity class $\mathcal{C}$, let $\exists^{\mathrm{p}} \cdot \mathcal{C}$ denote the class of languages L such that there exists a polynomial p and a $B \in \mathcal{C}$ such that for all x, $x \in L \iff \exists y\big(|y| \leq p(|x|), (x, y) \in B\big)$.

Unless stated otherwise, all hardness- and completeness-results are in terms of logspace many-one reducibility.

2.1 Satisfiability Problems for Circuits over Sets of Natural Numbers

We define the circuit model and related decision problems. A *circuit* $C = (V, E, g_C)$ is a finite, non-empty, directed, acyclic graph (V, E) with a specified node $g_C \in V$. The graph can contain multi-edges, it does not have to be connected, and $V = \{1, 2, \ldots, n\}$ for some $n \in \mathbb{N}$. Moreover, the nodes in the graph (V, E) are topologically ordered, i.e., for all $v_1, v_2 \in V$, if $v_1 < v_2$, then there is no path from v_2 to v_1. The nodes in V are also called *gates*. Nodes with indegree 0 are called *input gates* and g_C is called the *output gate*. If in a circuit there is an edge from gate u to gate v, then we say that u is a *direct predecessor* of v and v is the *direct successor* of u. If there is a path from u to v but u is not a direct predecessor of v, then u is an *indirect predecessor* of v and v is an *indirect successor* of u.

Let $\mathcal{O} \subseteq \{\cup, \cap, {}^-, +, \times\}$. An *$\mathcal{O}$-circuit with unassigned inputs* $C = (V, E, g_C, \alpha)$ is a circuit (V, E, g_C) whose gates are labeled by the labeling function $\alpha : V \to \mathcal{O} \cup \mathbb{N} \cup \{\star\}$ such that the following holds: Each gate has an indegree in $\{0, 1, 2\}$, gates with indegree 0 have labels from $\mathbb{N} \cup \{\star\}$, gates with indegree 1 have label ${}^-$, and gates with indegree 2 have labels from $\{\cup, \cap, +, \times\}$. Input gates with a label from $\mathbb{N}$ are called *assigned* (or constant) input gates; input gates with label $\star$ are called *unassigned* (or variable) input gates.

Let $u_1 < \cdots < u_n$ be the unassigned inputs in C, and let $x_1, \ldots, x_n \in \mathbb{N}$. By assigning value x_i to the input u_i for $1 \le i \le n$, we obtain an *$\mathcal{O}$-circuit* $C(x_1, \ldots, x_n)$ whose input gates are all assigned. Consequently, if C has no unassigned inputs, then $C = C()$.

As all input gates of the circuit $C(x_1, \ldots, x_n)$ have some natural number assigned to it, each gate $g \in V$ computes a set $I(g) \subseteq \mathbb{N}$, inductively defined as follows:

- If g is an input gate, then $I(g) \overset{df}{=} \begin{cases} \{\alpha(g)\}, & \text{if } \alpha(g) \neq \star, \\ \{x_k\}, & \text{if } g = u_k \text{ for a } k \in \{1, \ldots, n\}. \end{cases}$
- If g has label ${}^-$ and direct predecessor g_1, then $I(g) \overset{df}{=} \mathbb{N} - I(g_1)$.
- If g has label $\circ \in \{\cup, \cap, +, \times\}$ and direct predecessors g_1 and g_2, then we define $I(g) \overset{df}{=} I(g_1) \circ I(g_2)$.

We define $I(C(x_1, \ldots, x_n)) \overset{df}{=} I(g_C)$, the set computed by the $\mathcal{O}$-circuit $C(x_1, \ldots, x_n)$. If a circuit computes a singleton, we will sometimes write $I(C(x_1, \ldots, x_n)) = a$ instead of $I(C(x_1, \ldots, x_n)) = \{a\}$.

Definition 1. *Let $\mathcal{O} \subseteq \{\cup, \cap, {}^-, +, \times\}$. We define* membership *problems,* equivalence *problems, and* satisfiability *problems for circuits.*

$$\begin{aligned} \mathrm{MC}(\mathcal{O}) &\overset{df}{=} \{(C, b) \mid C \textit{ is an } \mathcal{O}\textit{-circuit without unassigned inputs, } b \in \mathbb{N}, \textit{ and } b \in I(C())\} \\ \mathrm{SC}(\mathcal{O}) &\overset{df}{=} \{(C, b) \mid C \textit{ is an } \mathcal{O}\textit{-circuit with unassigned inputs } u_1 < \cdots < u_n,\ b \in \mathbb{N}, \\ &\qquad \textit{and there exist } x_1, \ldots, x_n \in \mathbb{N} \textit{ such that } b \in I\big(C(x_1, \ldots, x_n)\big)\} \end{aligned}$$

When an $\mathcal{O}$-circuit $C = (V, E, g_c, \alpha)$ is used as input for an algorithm, then we use a suitable encoding such that it is possible to verify in deterministic logarithmic space whether a given string encodes a valid circuit. In the following, we will therefore assume that all algorithms start with such a validation of their input strings.

2.2 Examples

Let C be the circuit in Fig. 1(a). The $\star$ indicates that the sole input gate is unassigned. Moreover, we assume that the $\cap$-gate is the output gate. If 0 is assigned to the input gate, then both the input gate and the $+$-gate compute the set $\{0\}$. Consequently, the $\cap$-gate computes $\{0\}$. For all other assignments to the input gate, the circuit computes $\emptyset$. Hence, $(C, 0) \in \mathrm{SC}(\cap, +)$ and $(C, b) \notin \mathrm{SC}(\cap, +)$ for all $b \neq 0$.

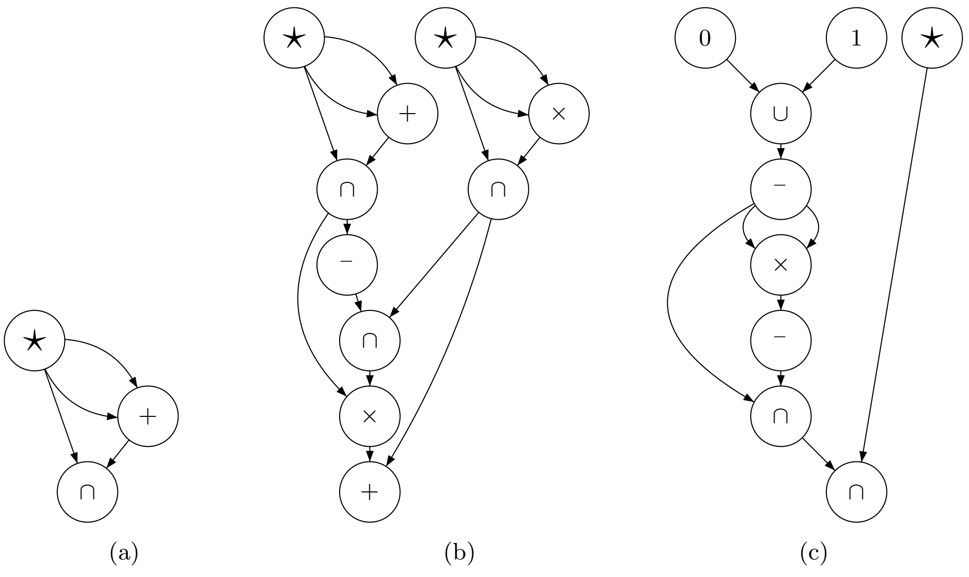

Fig. 1.

Let D be the circuit in Fig. 1(b). Depending on the assignments of the input gates, D computes either $\{1\}$ or $\emptyset$. Consequently, $(D, 1) \in \mathrm{SC}(^-, \cap, +, \times)$ and $(D, b) \notin \mathrm{SC}(^-, \cap, +, \times)$ for all $b \neq 1$. The example in Fig. 1(c) shows a circuit that generates either the empty set or any single prime.

3 Bounds that can be translated from MC($\mathcal{O}$) to SC($\mathcal{O}$)

This section summarizes upper and lower bounds that can be easily obtained from known results about membership problems. Here we can directly take over the lower bounds, since satisfiability problems are generalizations of membership problems. Moreover, we show that for sets of operations $\mathcal{O} \subseteq \{\cup, \cap, ^-, +\}$ and $\mathcal{O} \subseteq \{\cup, +, \times\}$, the satisfiability problem can be expressed as a *polynomially bounded* projection of the corresponding membership problem. This allows us to easily translate several known results into upper bounds for satisfiability problems.

Proposition 1. *The following results are immediate consequences of the results by McKenzie and Wagner [MW03].*

1. $\mathrm{SC}(^-, \cup, \cap, +)$, $\mathrm{SC}(\cup, \cap, +)$, $\mathrm{SC}(\cup, \cap, \times)$, $\mathrm{SC}(^-, \cup, \cap, \times)$, $\mathrm{SC}(\cup, +, \times)$ *are* $\leq_{\mathrm{m}}^{\log}$*-hard for* PSPACE.
2. $\mathrm{SC}(\cup, \times)$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* NP.
3. $\mathrm{SC}(\cap)$ *and* $\mathrm{SC}(\cup)$ *are* $\leq_{\mathrm{m}}^{\log}$*-complete for* NL.
4. $\mathrm{SC}(\times)$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* NL.
5. $\mathrm{SC}(\cup, \cap)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* P.

By definition, the problem $\mathrm{SC}(\mathcal{O})$ is an unrestricted projection of the $\mathrm{MC}(\mathcal{O})$ problem. We now show that for $\mathcal{O} \subseteq \{\cup, \cap, ^-, +\}$ and $\mathcal{O} \subseteq \{\cup, +, \times\}$ this projection is polynomially bounded.

Lemma 1. *Let C be a circuit over the operations $\mathcal{O} \subseteq \{\cup, \cap, ^-, +, \times\}$ with exactly n unassigned inputs. For any $b \in \mathbb{N}$, $x_1, \ldots, x_n \in \mathbb{N}$ and $c \leq b$ it holds that*

1. *if $\mathcal{O} \subseteq \{\cup, \cap, ^-, +\}$, then*
$$c \in I(C(x_1, \ldots, x_n)) \iff c \in I(C(\min(x_1, b+1), \ldots, \min(x_n, b+1))).$$
2. *if $\mathcal{O} \subseteq \{\cup, +, \times\}$, then*
$$c \in I(C(x_1, \ldots, x_n)) \Longrightarrow c \in I(C(\min(x_1, b+1), \ldots, \min(x_n, b+1))).$$

Proof. We show both parts by induction over the number of direct and indirect predecessors of the output gate and, for the sake of brevity, use the notations $\tilde{x} \overset{df}{=} x_1, x_2, \ldots, x_n$ and $\min(\tilde{x}, b+1) \overset{df}{=} \min(x_1, b+1), \min(x_2, b+1), \ldots, \min(x_n, b+1)$.

1. For the induction base consider C to be a circuit where the output gate is the first input gate and let $c \leq b$. If this input gate is assigned, the assertion is obviously true. Otherwise, the following equivalence holds:

$$c \in I(C(\tilde{x})) \iff c = x_1 \iff c = \min(x_1, b+1) \iff c \in I(C(\min(\tilde{x}, b+1)))$$

For the induction step, let C be a circuit whose output gate g has at least one direct predecessor, let $b \in \mathbb{N}$ and $c \leq b$. The cases where g has the label $^-$, $\cup$ or $\cap$ are straightforward. So let g have the label $+$ and the direct predecessors u and v. Let C_1 (resp., C_2) be the circuit that has u (resp., v) as output gate and is otherwise equal to C.

Note that $c \in I(C(\tilde{x}))$ if and only if there exist $s \in I(C_1(\tilde{x}))$ and $t \in I(C_2(\tilde{x}))$ such that $s, t \leq c \leq b$ and $s + t = c$. Thus, by the induction hypothesis, for all these s, t it holds that $s \in I(C_1(\tilde{x}))$ and $t \in I(C_2(\tilde{x}))$ if and only if $s \in I(C_1(\min(\tilde{x}, b+1)))$ and $t \in I(C_2(\min(\tilde{x}, b+1)))$. We obtain $c \in I(C(\tilde{x}))$ if and only if $c \in I(C_1(\min(\tilde{x}, b+1)) + C_2(\min(\tilde{x}, b+1))) = I(C(\min(\tilde{x}, b+1)))$.

2. The induction base, i.e., the case where the output gate is an input gate, is easy to see.

For the induction step, let C be a circuit whose output gate g has at least one direct predecessor, let $b \in \mathbb{N}$ and $c \leq b$. The case where g has label $\cup$ or $+$ is the same as in the proof of the first part and is thus omitted.

So let g have label $\times$, direct predecessors u and v and let C_1 (resp., C_2) be the circuit that has output gate u (resp., v) and is otherwise equal to C. If $c \in I(C(\tilde{x}))$, then there exist $s \in I(C_1(\tilde{x}))$ and $t \in I(C_2(\tilde{x}))$ such that $s \cdot t = c$.

In the case $c = 0$, without loss of generality, s must be zero. By the induction hypothesis we have $s \in I(C_1(\min(\tilde{x}, b+1)))$. Then, the only possibility for $c \notin I(C(\min(\tilde{x}, b+1)))$ is that $I(C_2(\min(\tilde{x}, b+1)))$ is the empty set. But this cannot happen, since the operations $+, \times$ and $\cup$ only produce empty sets if one of their inputs is the empty set and we start with singleton sets at the input gates. So if $c = 0 \in I(C(\tilde{x}))$, then also $c \in I(C(\min(\tilde{x}, b+1)))$.

The case $c \neq 0$ is analogous to the argumentation for $+$ in the first part and thus the induction is complete. □

Corollary 1. *Let C be a circuit over the operations $\mathcal{O} \subseteq \{\cup, \cap, ^-, +\}$ or $\mathcal{O} \subseteq \{\cup, +, \times\}$ with exactly n unassigned inputs and let $b \in \mathbb{N}$.*

$$(C, b) \in \mathrm{SC}(\mathcal{O}) \iff \exists x_1, \dots, x_n \in \{0, 1, \dots, b+1\} \text{ s.t. } (C(x_1, \dots, x_n), b) \in \mathrm{MC}(\mathcal{O})$$

Proof. From Lemma 1 we know that if $x_1, \dots, x_n$ is a satisfying input assignment for C, then so is $\min(x_1, b+1), \dots, \min(x_n, b+1)$. So we have a satisfying input assignment if and only if we have one whose values are bounded by $b+1$. Therefore, $(C, b) \in \mathrm{SC}(\mathcal{O})$ if and only if $b \in I(C(y_1, \dots, y_n))$ for some $y_1, \dots, y_n$ bounded by $b+1$. □

Corollary 2. *Let $\mathcal{O} \subseteq \{\cup, \cap, ^-, +\}$ or $\mathcal{O} \subseteq \{\cup, +, \times\}$ be a set of operations and let $\mathcal{C}$ be a complexity class. Then the following holds:*

$$\mathrm{MC}(\mathcal{O}) \in \mathcal{C} \implies \mathrm{SC}(\mathcal{O}) \in \exists^{\mathrm{p}} \cdot \mathcal{C}.$$

Proof. Follows from Corollary 1, since $|(x_1, \dots, x_n)|$ is polynomially bounded by $|(C, b)|$. □

Corollary 3. *It holds that*

1. $\mathrm{SC}(^-, \cup, \cap, +)$, $\mathrm{SC}(\cup, \cap, +)$, *and* $\mathrm{SC}(\cup, +, \times)$ *are in* PSPACE.
2. $\mathrm{SC}(^-, \cup, \cap)$, $\mathrm{SC}(\cap, +)$, $\mathrm{SC}(\cup, \times)$, $\mathrm{SC}(\cup, +)$, $\mathrm{SC}(+)$, *and* $\mathrm{SC}(+, \times)$ *are in* NP.

Proof. Since $\exists^{\mathrm{p}} \cdot \mathrm{PSPACE} = \mathrm{PSPACE}$ and $\exists^{\mathrm{p}} \cdot \mathrm{NP} = \mathrm{NP}$, the statement follows from Corollary 2 and the results by McKenzie and Wagner [MW03]. □

4 Satisfiability and Diophantine Equations

Circuits with gates $+$ and $\times$ can be used to compute multivariate polynomials. The presence of $\cap$ then allows us to translate the solvability of diophantine equations into the satisfiability of circuits. Hence the latter satisfiability problems are undecidable. Particularly, they are not polynomially bounded projections of their membership problems.

Lemma 2. *There exists a logspace computable function that on input of a multivariate polynomial $p(x_1,\dots,x_n)$ computes a $\{+,\times\}$-circuit C with n unassigned inputs such that for all $y_1,\dots,y_n \in \mathbb{N}$, $I(C(y_1,\dots,y_n)) = \{p(y_1,\dots,y_n)\}$.*

Proof. It suffices to argue for terms of the form x^e where $e \geq 1$. We construct a circuit C that has one unassigned input such that for all x, $I(C(x)) = \{x^e\}$. Let $\mathrm{bin}(e) = e_k \cdots e_0$. We start with a circuit that consists of gates $v_0,\dots,v_k$ such that v_0 is an input gate and all other gates have label $\times$. Moreover, for $i \in [0,k-1]$, there are two edges from gate v_i to gate v_{i+1}. Observe that $I(v_i) = I(v_0)^{2^i}$ and therefore,

$$\prod_{i,e_i=1} I(v_i) = I(v_0)^e.$$

This product can be produced by adding at most $k-1$ additional gates with label $\times$ to our circuit and by suitably connecting these new gates to the gates v_i where $e_i = 1$. We obtain a circuit C such that for all x, $I(C(x)) = \{x^e\}$. □

Theorem 1. $\mathrm{SC}(\cap,+,\times)$ *is undecidable.*

Proof. We show that the question of whether a given diophantine equation has solutions in $\mathbb{N}$ can be reduced to $\mathrm{SC}(\cap,+,\times)$. By the Davis-Putnam-Robinson-Matiyasevich theorem [DPR61,Mat70] this implies the undecidability of $\mathrm{SC}(\cap,+,\times)$.

Let $p(x_1,\dots,x_n) = 0$ be a diophantine equation with integer coefficients. By moving negative monoms and constants to the right-hand side, we obtain an equation

$$l(x_1,\dots,x_n) = r(x_1,\dots,x_n)$$

such that all coefficients in l, and all coefficients in r are positive. According to Lemma 2, we construct circuits C_l and C_r such that $C_l(x_1,\dots,x_n) = \{l(x_1,\dots,x_n)\}$ and $C_r(x_1,\dots,x_n) = \{r(x_1,\dots,x_n)\}$. Define a new circuit by

$$C'(x_1,\dots,x_n) \overset{df}{=} 0 \times (C_l(x_1,\dots,x_n) \cap C_r(x_1,\dots,x_n)).$$

Note that $p(x_1,\dots,x_n) = 0$ has a solution in $\mathbb{N}$ if and only if $(C',0) \in \mathrm{SC}(\cap,+,\times)$. □

5 Decidable Satisfiability Problems

In this section we prove upper and lower bounds for decidable satisfiability problems for circuits. Here it turns out that the problems $\mathrm{SC}(\cap,\times)$, $\mathrm{SC}(+)$, and $\mathrm{SC}(\times)$ are particularly interesting. For $\mathrm{SC}(\cap,\times)$, proving membership in NP is nontrivial. We finally prove this with help of certain systems of monom equations and the (also nontrivial) result that integer programming belongs to NP. Moreover, we show that $\mathrm{SC}(+)$ is likely to be more difficult than $\mathrm{SC}(\times)$. While $\mathrm{SC}(+)$ is NP-hard, $\mathrm{SC}(\times)$ belongs to $\mathrm{UP} \cap \mathrm{coUP}$.

5.1 Circuits with both Arithmetic and Set Operations

The problem SC($\cap, \times$) has an interesting property. In contrast to most other NP-complete problems, here proving the membership in NP is more difficult than proving the hardness for NP. We start working towards a proof for SC($\cap, \times$) $\in$ NP and define the following problem which asks for the solvability of *systems of monom equations*.

Name: MonEq
Instance: A list of equations of the following form.

$$\begin{aligned} x^5 z^7 &= 5^9 y^3 z^2 \\ y z^2 &= 2^3 x^5 \\ x^2 y^4 z^3 &= 3^{11} \end{aligned}$$

Question: Has this system of equations a solution within the natural numbers?

Formally, the problem MonEq is defined as follows (where we define 0^0 to be 1).

$$\begin{aligned} \text{MonEq} \overset{df}{=} \{(A,B,C,D) \,\big|\, & A = (a_{i,j}) \in \mathbb{N}^{m \times n},\ B = (b_{i,j}) \in \mathbb{N}^{m \times n},\ C = (c_1, \dots, c_m) \in \mathbb{N}^m, \\ & D = (d_1, \dots, d_m) \in \mathbb{N}^m, \text{ and there exist } x_1, \dots, x_n \in \mathbb{N} \text{ such} \\ & \text{that for all } i \in [1,m],\ \textstyle\prod_{j=1}^{n} x_j^{a_{i,j}} = c_i^{d_i} \cdot \prod_{j=1}^{n} x_j^{b_{i,j}}\} \end{aligned}$$

Note that formally, this definition neither allows constant factors at the left-hand side of equations nor allows products of constant factors like $2^{91} \cdot 3^{93} \cdot 5^{97}$. However, such factors can be easily expressed by using additional variables. For example, the equation

$$7^3 \cdot 15^{70} \cdot x^5 y^7 = 3^7 z^3$$

can be equivalently transformed into the following system.

$$\begin{aligned} a &= 7^3 \\ b &= 15^{70} \\ a b x^5 y^7 &= 3^7 z^3 \end{aligned}$$

We show that systems of monom equations can be solved in nondeterministic polynomial time. Our proof transforms the original problem MonEq to a restricted version MonEq$'$ and further to a more restricted version MonEq$''$. Then we show the latter to be in NP where we use the fact that integer programming belongs to NP.

Lemma 3. MonEq $\in$ NP.

Proof. We start with the definition of a variant of MonEq that restricts to positive constant factors and positive solutions.

$$\begin{aligned} \text{MonEq}' \overset{df}{=} \{(A,B,C,D) \,\big|\, & A = (a_{i,j}) \in \mathbb{N}^{m \times n},\ B = (b_{i,j}) \in \mathbb{N}^{m \times n},\ C = (c_1, \dots, c_m) \in (\mathbb{N}^+)^m, \\ & D = (d_1, \dots, d_m) \in \mathbb{N}^m, \text{ and there exist } x_1, \dots, x_n \in \mathbb{N}^+ \text{ such that} \\ & \text{for all } i \in [1,m],\ \textstyle\prod_{j=1}^{n} x_j^{a_{i,j}} = c_i^{d_i} \cdot \prod_{j=1}^{n} x_j^{b_{i,j}}\} \end{aligned}$$

Assume for the moment that we have shown $\text{MonEq}' \in \text{NP}$. Under this assumption we can describe a nondeterministic polynomial-time algorithm that accepts MonEq. The input is a MonEq instance (A, B, C, D). First, we nondeterministically guess the variables that will be equal to 0, i.e., we guess a set $I \subseteq \{1, \dots, n\}$ and demand that $x_i = 0$ for all $i \in I$ and that $x_j > 0$ for all $j \notin I$. This allows us to determine whether or not a certain side of an equation is 0. We delete all equations whose sides both equal 0. If there exists an equation that equals 0 on one side and that is different from 0 on the other side, then this equation cannot be satisfied and hence we reject. Otherwise, all remaining equations are greater than 0 on both sides. Hence we arrived at a MonEq' instance which by assumption can be solved in NP. This shows $\text{MonEq} \in \text{NP}$. So it remains to prove $\text{MonEq}' \in \text{NP}$.

Let us define a variant of MonEq' that restricts to the case where all constant factors and all components of the solution are powers of the same prime number p.

$$\begin{aligned}\text{MonEq}'' \overset{\mathit{df}}{=} \{(A, B, p, D) \mid{} & A = (a_{i,j}) \in \mathbb{N}^{m \times n},\ B = (b_{i,j}) \in \mathbb{N}^{m \times n},\ D = (d_1, \dots, d_m) \in \mathbb{N}^m, \\ & p \text{ is a prime, and there exist } x_1, \dots, x_n \in \{p^r \mid r \in \mathbb{N}\} \text{ such that} \\ & \text{for all } i \in [1, m],\ \textstyle\prod_{j=1}^n x_j^{a_{i,j}} = p^{d_i} \cdot \prod_{j=1}^n x_j^{b_{i,j}}\}\end{aligned}$$

Assume for the moment that we have shown $\text{MonEq}'' \in \text{NP}$. Under this assumption we show that $\text{MonEq}' \in \text{NP}$. Let the MonEq' instance (A, B, C, D) be our input and let $p_1, \dots, p_l$ be the primes that appear in the prime factorization of the numbers $c_1, \dots, c_m$. Moreover, for $k \in [1, l]$ and $i \in [1, m]$, let $e_{k,i}$ denote the exponent of the prime factor p_k in the factorization of c_i. Let us observe the following equivalence.

$$\begin{aligned}(A, B, C, D) \in \text{MonEq}' \quad \iff \quad & \forall k \in [1, l]\ \ \exists x_{k,1}, \dots, x_{k,n} \in \{p_k^r \mid r \in \mathbb{N}\}\ \ \forall i \in [1, m], \\ & \textstyle\prod_{j=1}^n x_{k,j}^{a_{i,j}} = p_k^{d_i} \cdot \prod_{j=1}^n x_{k,j}^{b_{i,j}}\end{aligned} \tag{1}$$

The implication from right to left is easy to see, since with

$$x_j \overset{\mathit{df}}{=} \prod_{k=1}^{l} x_{k,j}$$

we obtain a solution $x_1, \dots, x_n$ for the MonEq' instance (A, B, C, D). So let us consider the implication from left to right and let $x_1, \dots, x_n$ be a solution for (A, B, C, D). Let $x_{k,j}$ be the number that is obtained from x_j by removing all prime factors different from p_k. Observe that $x_{k,1}, \dots, x_{k,n}$ is a solution for the system on the right-hand side of (1). This proves (1).

The equivalence (1) shows that MonEq' is conjunctively truth-table reducible to MonEq''. By our assumption, the latter is in NP and therefore, we obtain $\text{MonEq}' \in \text{NP}$. So it remains to prove $\text{MonEq}'' \in \text{NP}$.

The definition of MonEq'' demands that each element x_j of the solution can be written as $x_j = p^{e_j}$ for a suitable $e_j \in \mathbb{N}$. We obtain

$$\begin{aligned}(A, B, p, D) \in \text{MonEq}'' \iff{} & p \text{ is prime and there exist } e_1, \dots, e_n \in \mathbb{N} \text{ such that} && (2) \\ & \text{for all } i \in [1, m],\ \textstyle\prod_{j=1}^n p^{e_j a_{i,j}} = p^{d_i} \cdot \prod_{j=1}^n p^{e_j b_{i,j}} \\ \iff{} & p \text{ is prime and there exist } e_1, \dots, e_n \in \mathbb{N} \text{ such that} && (3) \\ & \text{for all } i \in [1, m],\ \textstyle\sum_{j=1}^n e_j a_{i,j} = d_i + \sum_{j=1}^n e_j b_{i,j}.\end{aligned}$$

The right-hand side of (3) can be expressed by the following integer program.

- $\sum_{j=1}^{n} e_j a_{i,j} \leq d_i + \sum_{j=1}^{n} e_j b_{i,j}$ for $i \in [1, m]$
- $\sum_{j=1}^{n} e_j a_{i,j} \geq d_i + \sum_{j=1}^{n} e_j b_{i,j}$ for $i \in [1, m]$
- $e_j \geq 0$ for $j \in [1, n]$

For such systems of inequalities, the existence of integer solutions can be verified in NP [Kar72]. This shows MonEq$''$ $\in$ NP and finishes the proof of the lemma. □

Using the fact that systems of monom equations can be solved in nondeterministic polynomial time we now show that SC$(\cap, \times)$ belongs to NP.

Theorem 2. SC$(\cap, \times) \in$ NP

Proof. We describe a nondeterministic polynomial-time algorithm for SC$(\cap, \times)$ on input (C, d). Without loss of generality we may assume that the nodes $1, \ldots, m$ are the unassigned input gates and the nodes $m+1, \ldots, m+n$ are the assigned input gates with labels $b_1, \ldots, b_n$. We recursively attach monoms of the form $x_1^7 x_2^{23} \cdots x_{m+n}^5$ to the gates of C: We attach the monom x_i to input gate i. Let i be a gate with the direct predecessors i_1 and i_2 such that the monom M_1 is attached to i_1 and M_2 is attached to i_2. If i is a $\times$-gate, then we attach the monom $M_1 \cdot M_2$ to i (where we simplify the product in the sense that multiple occurrences of variables x_j are combined). If i is a $\cap$-gate, then we attach the monom M_1 to i. In this way, we attach a monom to each gate of C. Now each $\cap$-gate i that is (directly or indirectly) connected to the output gate induces a monom equation $M_1 = M_2$ where M_1 and M_2 are the monoms that are attached to i's direct predecessors. These equations form a system of monom equations. Next we add the following equations to this system.

- For $i \in [1, n]$ the equation $x_{m+i} = b_i$ where b_i is the label of the assigned input gate $m + i$.
- The equation $M = d$ where M is the monom attached to the output gate.

Our algorithm accepts if and only if the obtained system of monom equations has a solution within the natural numbers.

By Lemma 3, the described algorithm is a nondeterministic polynomial-time algorithm. So it remains to argue for the correctness of this algorithm.

For a monom M attached to some gate, let $M(a_1, \ldots, a_m, b_1, \ldots, b_n)$ denote the number that is obtained when M is evaluated for $x_1 = a_1, \ldots, x_m = a_m, x_{m+1} = b_1, \ldots, x_{m+n} = b_n$. A straightforward induction on the structure of C yields the following.

Claim. If gate g has the monom M attached, then for all $a_1, \ldots, a_m \in \mathbb{N}$, the gate g of the circuit $C(a_1, \ldots, a_m)$ either computes $\emptyset$ or computes the set $\{M(a_1, \ldots, a_m, b_1, \ldots, b_n)\}$.

We show that the algorithm accepts (C, d) if and only if $(C, d) \in$ SC$(\cap, \times)$.

Assume our algorithm accepts on input (C, d). So there exist $a_1, \ldots, a_m$ such that $a_1, \ldots, a_m, b_1, \ldots, b_n$ is a solution for the constructed system of monom equations. Suppose $I(C(a_1, \ldots, a_m)) = \emptyset$. Then there exists a $\cap$-gate g with direct predecessors g_1

and g_2 such that g is connected to the output gate, $I(g_1) \neq \emptyset$, $I(g_2) \neq \emptyset$, and $I(g_1) \neq I(g_2)$. Let M, M_1, and M_2 be the monoms attached to g, g_1, and g_2 respectively. By Claim 5.1, $I(g_1) = \{M_1(a_1, \ldots, a_m, b_1, \ldots, b_n)\}$ and $I(g_2) = \{M_2(a_1, \ldots, a_m, b_1, \ldots, b_n)\}$. The equation $M_1 = M_2$ appears in our system of monom equations. Therefore, $M_1(a_1, \ldots, a_m, b_1, \ldots, b_n) = M_2(a_1, \ldots, a_m, b_1, \ldots, b_n)$ and hence $I(g_1) = I(g_2)$. We have already seen that the latter is not true and so it follows that $I(C(a_1, \ldots, a_m)) \neq \emptyset$. Now let M denote the monom attached to the output gate. By Claim 5.1, $I(C(a_1, \ldots, a_m)) = \{M(a_1, \ldots, a_m, b_1, \ldots, b_n)\}$. The equation $M = d$ appears in the system of monom equations. This shows $I(C(a_1, \ldots, a_m)) = \{d\}$ and hence $(C, d) \in \mathrm{SC}(\cap, \times)$.

Conversely, assume now that $(C, d) \in \mathrm{SC}(\cap, \times)$, i.e., there exist $a_1, \ldots, a_m \in \mathbb{N}$ such that $I(C(a_1, \ldots, a_m)) = \{d\}$. We show that $x_1 = a_1$, …, $x_m = a_m$, $x_{m+1} = b_1$, …, $x_{m+n} = b_n$ is a solution for the system of monom equations that is constructed by the algorithm. The latter immediately implies that the algorithm accepts on input (C, d). In the circuit $C(a_1, \ldots, a_m)$, each $\cap$-gate g that is connected to the output gate computes a nonempty set. So if g_1 and g_2 are the predecessors of g, then $I(g) = I(g_1) = I(g_2)$. Let M, M_1, and M_2 be the monoms attached to g, g_1, and g_2 respectively. From Claim 5.1 it follows that $M_1(a_1, \ldots, a_m, b_1, \ldots, b_n) = M_2(a_1, \ldots, a_m, b_1, \ldots, b_n)$. So all equations of the form $M_1 = M_2$ are satisfied. Moreover, the additional equations of the form $x_{m+i} = b_i$ are trivially satisfied by our solution. From $I(C(a_1, \ldots, a_m)) = \{d\}$ and from Claim 5.1 it follows that $M(a_1, \ldots, a_m, b_1, \ldots, b_n) = d$ where M is the monom attached to C's output gate. This shows that all equations of our system are satisfied by the solution $(a_1, \ldots, a_m, b_1, \ldots, b_n)$ and it follows that the algorithm accepts.

□

Finally, we establish the lower bound for $\mathrm{SC}(\cap, \times)$.

Theorem 3. $\mathrm{SC}(\cap, \times)$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* NP.

Proof. Now we show the NP-hardness of $\mathrm{SC}(\cap, \times)$ by describing a $\leq_{\mathrm{m}}^{\log}$-reduction from 3SAT to $\mathrm{SC}(\cap, \times)$. Let $f(x_1, \ldots, x_n)$ be the input, i.e., $f(x_1, \ldots, x_n) = d_1 \wedge d_2 \wedge \cdots \wedge d_m$ where the d_i are clauses that are a disjunction of three literals a_i, b_i, and c_i (literals are either x_i or $\overline{x_i}$). We construct a circuit C that has the following input gates:

- $x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}$
- $y_1, \overline{y_1}, \ldots, y_m, \overline{y_m}$ and $z_1, \overline{z_1}, \ldots, z_m, \overline{z_m}$

For each variable x_i we construct a subcircuit that computes $(x_i \times \overline{x_i}) \times (x_i \times \overline{x_i}) \times (x_i \times \overline{x_i})$ at its output gate s_i. For each clause $d_i = a_i \vee b_i \vee c_i$ where a_i, b_i, and c_i are literals, we construct three subcircuits: a first one that computes $(y_i \times \overline{y_i}) \times (y_i \times \overline{y_i}) \times (y_i \times \overline{y_i})$ at its output gate u_i, a second one that computes $(z_i \times \overline{z_i}) \times (z_i \times \overline{z_i}) \times (z_i \times \overline{z_i})$ at its output gate v_i, and a third one that computes $a_i \times b_i \times c_i \times y_i \times z_i$ at its output gate w_i. Finally, the output gate of C computes the intersection of all s_i, u_i, v_i, and w_i. Now our reduction outputs $(C, 8)$.

Observe that the reduction can be computed in logarithmic space. We now argue that it reduces $3\mathrm{SAT} \leq_{\mathrm{m}}^{\log} \mathrm{SC}(\cap, \times)$. First, assume that the input $f(x_1, \ldots, x_n) = d_1 \wedge d_2 \wedge \cdots \wedge d_m$

belongs to 3SAT. So there exist $e_1, \dots, e_n \in \{0,1\}$ such that $f(e_1, \dots, e_n) = 1$. We assign the numbers 2^{e_i} to the input gates x_i and 2^{1-e_i} to the input gates $\overline{x_i}$. By doing this, we make sure that all gates s_i evaluate to 8. Moreover, we assign the following numbers to the input gates y_i, $\overline{y_i}$, z_i, and $\overline{z_i}$: If the clause d_i contains 3 literals that evaluate to true with respect to the assignment $e_1, \dots, e_n$, then assign 1 to y_i and z_i and assign 2 to $\overline{y_i}$ and $\overline{z_i}$. If d_i contains 2 such literals, then assign 2 to y_i and $\overline{z_i}$ and assign 1 to $\overline{y_i}$ and z_i. If d_i contains 1 such literal (there must be at least one), then assign 2 to y_i and z_i and assign 1 to $\overline{y_i}$ and $\overline{z_i}$. Observe that this assignment makes sure that all gates u_i, v_i, and w_i evaluate to 8. Hence, the output gate of C evaluates to 8 which shows $(C, 8) \in \mathrm{SC}(\cap, \times)$.

Now assume that $(C, 8) \in \mathrm{SC}(\cap, \times)$; we will show $f(x_1, \dots, x_n) \in$ 3SAT. By assumption, all gates s_i, u_i, v_i, and w_i evaluate to 8. The value 8 at each s_i implies that one of x_i and $\overline{x_i}$ has input 2 while the other one has input 1. Similarly, the value 8 at each u_i (resp., v_i) implies that one of y_i and $\overline{y_i}$ (resp., z_i and $\overline{z_i}$) has input 2 while the other one has input 1. If x_i has input 2, then let $e_i = 1$, otherwise let $e_i = 0$. The value 8 at each w_i shows that $a_i \times b_i \times c_i \times y_i \times z_i$ evaluates to 8. Since y_i and z_i are either 1 or 2, at least one of a_i, b_i, or c_i must evaluate to 2. Hence, the clause d_i contains at least one literal that evaluates to true with respect to the assignment $e_1, \dots, e_n$. This shows $f(x_1, \dots, x_n) \in$ 3SAT. Therefore, 3SAT$\leq_{\mathrm{m}}^{\log}\mathrm{SC}(\cap, \times)$ and hence $\mathrm{SC}(\cap, \times)$ is $\leq_{\mathrm{m}}^{\log}$-hard for NP. □

The next corollary shows that we can utilize the algorithm presented in Theorem 2 which evaluates $\{\cap, \times\}$-circuits also to evaluate $\{\cup, \cap, \times\}$-circuits: However, to cope with the $\cup$-gates we first have to unfold the circuit such that no inner gate has outdegree greater than 1. This can cause an exponential blow up in the size of the circuit.

Corollary 4. $\mathrm{SC}(\cup, \cap, \times) \in$ NEXP.

Proof. Let C be a $\{\cup, \cap, \times\}$-circuit and $b \in \mathbb{N}$. We describe a nondeterministic exponential time algorithm that decides $(C, b) \in \mathrm{SC}(\cup, \cap, \times)$. In exponential time, we can unfold the circuit into a (possibly exponentially larger) circuit C' where all gates except from the variable input gates have outdegree at most 1: Starting with the output gate, for each gate g with direct successors $s_1, \dots, s_k$, we replace g with identical gates $g_1, \dots, g_k$ such that the direct successor of gate g_i is s_i. Each g_i has the same predecessors that g had, so we now have k copies of all predecessors of g. We then repeat this procedure for the next gate with outdegree greater than one. We exclude the variable input gates from this procedure such that we do not get any new copies of these in the circuit C'.

Observe that C' is just a less space efficient representation of C. It has the same unassigned inputs and it holds for all $x_1, \dots, x_n \in \mathbb{N}$ that

$$I(C(x_1, \dots, x_n)) = I(C'(x_1, \dots, x_n)).$$

Observe that by unfolding the circuit first, we have ensured that it suffices to consider one natural number per gate when evaluating the unfolded circuit. For $\cup$-gates, we nondeterministically guess whether we take the left or right predecessor: For each $\cup$-gate g in C', nondeterministically guess one bit $b_g \in \{0,1\}$. Evaluate the circuit using the same algorithm as in Theorem 2. Whenever the algorithm encounters a $\cup$-gate g, continue with the left predecessor of g if $b_g = 0$ and continue with the right predecessor if $b_g = 1$. As the algorithm presented in Theorem 2 is a nondeterministic polynomial time algorithm, the above steps can be performed in time $2^{p(|C|)}$ for a suitable polynomial p. This proves $\mathrm{SC}(\cup,\cap,\times) \in \mathrm{NEXP}$. □

5.2 Circuits with either Arithmetic or Set Operations.

We now discuss that $\mathrm{SC}(\times)$ is easier than $\mathrm{SC}(+)$ unless NP = coNP. More precisely, we show that $\mathrm{SC}(\times) \in \mathrm{UP} \cap \mathrm{coUP}$ and prove $\mathrm{SC}(+)$ to be NP-complete. Here it is interesting to note that the same variant of the KNAPSACK-problem is used to establish both, the upper bound for $\mathrm{SC}(\times)$ and the lower bound for $\mathrm{SC}(+)$. The latter requires a version of KNAPSACK that allows the repeated use of weights. The upper bound for $\mathrm{SC}(\times)$ depends on the property that KNAPSACK is *weakly* NP-complete [GJ79], i.e., the problem is easy to solve if the weights are given in unary representation. These constraints lead to the following variant of the KNAPSACK-problem which is known to be weakly NP-complete [Pap94, 9.5.33].

$$\mathrm{KNAPSACK}' \overset{df}{=} \{(v_1,\dots,v_n,b) \mid n \geq 0,\ v_1,\dots v_n, b \in \mathbb{N} \text{ and there exist } u_1,\dots,u_n \in \mathbb{N} \text{ such that } \textstyle\sum_{i=1}^{n} u_i v_i = b\}$$

Theorem 4. $\mathrm{SC}(+)$ *and* $\mathrm{SC}(+,\times)$ *are* $\leq_{\mathrm{m}}^{\log}$*-complete for* NP.

Proof. We describe a reduction from $\mathrm{KNAPSACK}'$ to $\mathrm{SC}(+)$. On input $(v_1,\dots,v_n,b)$, if some v_i equals 0, then by deleting v_i we obtain an equivalent but shorter instance of the problem. If $n = 0$, then the problem is easy to solve. So we may assume that $n \geq 1$ and that all v_i are greater than 0. We use the method described in the proof of Lemma 2 and construct in deterministic logarithmic space a $\{+\}$-circuit C such that $I(C(x_1,\dots,x_n)) = \{v_1x_1 + \dots + v_nx_n\}$. (In contrast to Lemma 2, here we do not need $\times$-gates, since the polynomial $v_1x_1+\dots+v_nx_n$ has degree 1.) The reduction finally outputs (C,b). Observe that $(v_1,\dots,v_n,b)$ if and only if $(C,b) \in \mathrm{SC}(+)$. So $\mathrm{SC}(+)$ and $\mathrm{SC}(+,\times)$ are NP-hard. Membership in NP follows from Corollary 3. □

By $\mathrm{MC}(\times) \in \mathrm{NL}$ [MW03] and Corollary 2, it is immediately clear that $\mathrm{SC}(\times) \in \mathrm{NP}$. We now prove the better upper bound $\mathrm{UP} \cap \mathrm{coUP}$ by utilizing dynamic programming. More precisely, we will show that testing whether $(C, p^e) \in \mathrm{SC}(\times)$ for a prime p and $e \geq 0$ reduces in polynomial time to solving a $\mathrm{KNAPSACK}'$ instance where the weights are encoded in unary. By the weak NP-completeness of $\mathrm{KNAPSACK}'$, the latter instance can be solved in polynomial time via dynamic programming. We then exploit this to prove that $\mathrm{SC}(\times) \in \mathrm{UP} \cap \mathrm{coUP}$.

Proposition 2 ([GJ79]). $\mathrm{KNAPSACK}'$ *is computable in polynomial time if the input numbers are given in unary coding.*

Theorem 5. $\mathrm{SC}(\times) \in \mathrm{UP} \cap \mathrm{coUP}$.

Proof. Let C be a $\{\times\}$-circuit with unassigned inputs $u_1, \ldots, u_k$ and let $n \geq 0$. We now describe how to decide whether $(C, n) \in \mathrm{SC}(\times)$. Recall that $\mathrm{MC}(\times) \in \mathrm{NL}$ [MW03], hence a circuit without unassigned inputs can be evaluated in polynomial time. If $n = 0$, we accept if and only if $I(C(0, 0, \ldots, 0)) = 0$. If $n > 0$, we compute $a \overset{df}{=} I(C(1, 1, \ldots, 1))$. In the case $a = 0$ we reject, since $a = 0$ implies that the circuit computes 0 regardless of the inputs. If $a \neq 0$, then no constant input that is connected to the output node can be labeled with 0. In addition, we can conclude that every number computable by the circuit is divisible by a. Consequently, if n is not divisible by a, we reject.

Let C' be the circuit obtained by replacing all labels of constant input gates in C by 1. Clearly, this transformation can be performed in polynomial time. For all $b \geq 0$ it now holds that

$$(C, a \cdot b) \in \mathrm{SC}(\times) \iff (C', b) \in \mathrm{SC}(\times).$$

Set $n' \overset{df}{=} \frac{n}{a}$.

The following nondeterministic algorithm decides whether $(C', n') \in \mathrm{SC}(\times)$:

```
1. guess numbers $m, p_1, \ldots, p_m, e_1, \ldots, e_m$ such that $1 \leq m \leq |n'|$,
   $2 \leq p_1 < p_2 < \cdots < p_m \leq n'$, and for all i it holds that $1 \leq e_i \leq |n'|$
2. if at least one of the $p_i$ is not prime then reject
3. if $n' \neq p_1^{e_1} \cdots p_m^{e_m}$ then reject
4. // here $n' = p_1^{e_1} \cdots p_m^{e_m}$ is the prime factorization of $n'$
5. if $(C', p_i^{e_i}) \in \mathrm{SC}(\times)$ for all $i \in [1, m]$ then accept else reject
```

Step 2 is possible in polynomial time by the algorithm by Agrawal, Kayal, and Saxena [AKS04].

We now explain that step 5 can also be carried out in polynomial time. Note that there exist $e_1, \ldots, e_k$ such that for every assignment $x_1, \ldots, x_n$ to the input gates $u_1, \ldots, u_k$, it holds that

$$I(C'(x_1, \ldots, x_k)) = x_1^{e_1} \cdots x_k^{e_k}.$$

The exponents only depend on the circuit C'. Moreover, they can be computed in polynomial time: First transform C' into a $+$-circuit C'' as follows: Replace all $\times$-nodes with $+$-nodes. Then relabel all constant inputs with 0 instead of 1. Now observe that

$$I(C''(\underbrace{0, \ldots, 0}_{j-1}, 1, \underbrace{0, \ldots, 0}_{k-j}) = e_j.$$

As this can be done in polynomial time, we have shown that all exponents can be computed in polynomial time.

Claim. For a prime p and $e \geq 0$, $(C', p^e) \in \mathrm{SC}(\times)$ can be tested in polynomial time.

Proof. If a prime power p^e is computed at the output gate of C', then it follows that all input gates must have powers of p assigned to then. In this case it suffices to solve the following problem: Do there exist $y_1, \ldots, y_k$ such that $(p^{y_1})^{e_1} \cdots (p^{y_k})^{e_k} = p^e$?

We conclude that

$$(C', p^e) \in \mathrm{SC}(\times) \Leftrightarrow \exists y_1, \dots, y_k(e_1y_1 + e_2y_1 + \cdots + e_ky_k = e).$$

So it turns out that asking whether $(C', p^e) \in \mathrm{SC}(\times)$ is precisely the KNAPSACK′ problem. Since $e \leq \log n$, it follows that the unary coding of e is polynomial in n and hence polynomial in the input. By Proposition 2, it follows that we can check $(C, p^e) \in \mathrm{SC}(\times)$ in polynomial time. This proves the claim □

We have shown that the above algorithm runs in polynomial time. To see that the algorithm accepts if and only if $(C', n') \in \mathrm{SC}(\times)$, observe that the following holds:

$$(C', n') \in \mathrm{SC}(\times) \Leftrightarrow \forall_{1 \leq i \leq m}(C', p_i^{l_i}) \in \mathrm{SC}(\times),$$

where $n' = p_1^{l_1} \cdot \ldots \cdot p_m^{l_m}$ is the prime factorization of n'.

Every number has a unique prime factorization. Therefore, there exists exactly one path on which the algorithm reaches step 5. This shows $\mathrm{SC}(\times) \in \mathrm{UP}$. If we exchange ‘accept’ and ‘reject’ in step 5, then we arrive at an algorithm witnessing $\overline{\mathrm{SC}(\times)} \in \mathrm{UP}$. This completes the proof. □

We now show the NP-hardness of $\mathrm{SC}(^-, \cup, \cap)$ by reducing 3SAT to $\mathrm{SC}(^-, \cup, \cap)$. Here we utilize the natural correspondence between $\{^-, \cup, \cap\}$ and $\{\neg, \vee, \wedge\}$.

Theorem 6. $\mathrm{SC}(^-, \cup, \cap)$ *is* $\leq_\mathrm{m}^{\log}$*-complete for* NP.

Proof. By Corollary 3, $\mathrm{SC}(^-, \cup, \cap) \in \mathrm{NP}$. Moreover, $\mathrm{3SAT} \leq_\mathrm{m}^{\log} \mathrm{SC}(^-, \cup, \cap)$ by translating the Boolean operations $\neg, \vee, \wedge$ into the set operations $^-, \cup, \cap$ and by asking whether the resulting circuit can produce 1 (i.e., $A \subseteq \mathbb{N}$ is interpreted as *true* if and only if $1 \in A$). □

6 Conclusions

Table 1 summarizes our results. It shows that in most cases we can precisely characterize the complexity of the different variants of the satisfiability problem. Several open questions are apparent from it.

Our main open question is whether $\mathrm{SC}(^-, \cup, \cap, \times)$ is decidable. In the absence of +-gates, we cannot express general diophantine equations, which indicates the difficulty of proving undecidability. On the other hand, we do not know any decidable upper bound for this problem, since here the complementation-gates make it difficult to find a bound for the input gates. As the example in Fig. 1(c) shows, such circuits can express nontrivial statements about prime numbers. A further open question is to find a better lower bound for the satisfiability problem for $\{\times\}$-circuits. We prove this problem to be in $\mathrm{UP} \cap \mathrm{coUP}$. Membership in P seems to be difficult, since $\mathrm{SC}(\times)$ comprises the following factoring-like problem: Is the factorization of a given number n of a certain form, for instance $n = x^3 \cdot y^5 \cdot z^2$? However, proving $\mathrm{SC}(\times)$ to be hard for factorization is still open.

	SC			
$\mathcal{O}$	Lower Bound		Upper Bound	
$^- \cup \cap + \times$	undecidable			
$^- \cup \cap +$	PSPACE	Pr.1	PSPACE	Co.3
$^- \cup \cap \times$	PSPACE	Pr.1		
$^- \cup \cap$	NP	Th.6	NP	Co.3
$\cup \cap + \times$	undecidable			
$\cup \cap +$	PSPACE	Pr.1	PSPACE	Co.3
$\cup \cap \times$	PSPACE	Pr.1	NEXP	Co.4
$\cup \cap$	P	Pr.1	P	Pr.1
$\cup + \times$	PSPACE	Pr.1	PSPACE	Co.3
$\cup +$	NP	Th.4	NP	Co.3
$\cup \times$	NP	Pr.1	NP	Co.3
$\cup$	NL	Pr.1	NL	Pr.1
$\cap + \times$	undecidable			
$\cap +$	NP	Th.4	NP	Co.3
$\cap \times$	NP	Th.3	NP	Th.2
$\cap$	NL	Pr.1	NL	Pr.1
$+ \times$	NP	Th.4	NP	Th.4
$+$	NP	Th.4	NP	Th.4
$\times$	NL	Pr.1	$\mathrm{UP} \cap \mathrm{coUP}$	Th.5

Table 1. Upper and lower bounds for $\mathrm{SC}(\mathcal{O})$. All bounds are with respect to $\leq_{\mathrm{m}}^{\log}$-reductions and the numbers in parentheses refer to the corresponding theorems.

References

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. *Annals of Mathematics*, 160:781–793, 2004.

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings 3rd Symposium on Theory of Computing*, pages 151–158. ACM Press, 1971.

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

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Mathematical sciences series. Freeman, 1979.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

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

[MW03] P. McKenzie and K. W. Wagner. The complexity of membership problems for circuits over sets of natural numbers. In *Proceedings 20th Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *Lecture Notes in Computer Science*, pages 571–582. Springer Verlag, 2003.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[SM73] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *Proceedings 5th ACM Symposium on the Theory of Computing*, pages 1–9. ACM Press, 1973.

[Wag84] K. Wagner. The complexity of problems concerning graphs with regularities. In *Proceedings Mathematical Foundations of Computer Science*, volume 176 of *Lecture Notes in Computer Science*, pages 544–552. Springer-Verlag, 1984.

[Yan00] K. Yang. Integer circuit evaluation is PSPACE-complete. In *IEEE Conference on Computational Complexity*, pages 204–213, 2000.