Circuits and Expressions over Finite Semirings

MOSES GANARDI, DANNY HUCKE, DANIEL KÖNIG, and MARKUS LOHREY, University of Siegen, Germany

The computational complexity of the circuit and expression evaluation problem for finite semirings is considered, where semirings are not assumed to have an additive or a multiplicative identity. The following dichotomy is shown: If a finite semiring is such that (i) the multiplicative semigroup is solvable and (ii) it does not contain a subsemiring with an additive identity 0 and a multiplicative identity $1 \neq 0$, then the circuit evaluation problem is in $\mathsf{DET} \subseteq \mathsf{NC}^2$ and the expression evaluation problem for the semiring is in TC^0. For all other finite semirings, the circuit evaluation problem is P-complete and the expression evaluation problem is NC^1-complete. As an application we determine the complexity of intersection non-emptiness problems for given context-free grammars (regular expressions) with a fixed regular language.

CCS Concepts: • **Theory of computation → Circuit complexity**;

Additional Key Words and Phrases: semirings, circuit evaluation problem, expression evaluation

ACM Reference Format:
Moses Ganardi, Danny Hucke, Daniel König, and Markus Lohrey. 2018. Circuits and Expressions over Finite Semirings. *ACM Trans. Comput. Theory* 1, 1, Article 1 (January 2018), 29 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Circuit and expression evaluation problems are among the most well-studied computational problems in complexity theory. In its most general formulation, one has an algebraic structure $\mathcal{A} = (A, f_1, \dots, f_k)$ with operations $f_i \colon A^{n_i} \to A$ over some domain A. A circuit over $\mathcal{A}$ is a directed acyclic graph (dag) where every inner node is labelled with one of the operations f_i and has exactly n_i outgoing edges that are linearly ordered. The leaf nodes of the dag are labelled with elements of A (for this, one needs a suitable finite representation of elements from A), and there is a distinguished output node. The circuit evaluation problem for $\mathcal{A}$, short $\mathsf{CEP}(\mathcal{A})$, is to evaluate this dag in the natural way, and to return the value of the output node. Circuits can be seen as a succinct representation of expressions over the algebra. An expression over the structure $\mathcal{A}$ is a term built up from elements in A and the function symbols f_i. The expression evaluation problem for $\mathcal{A}$, short $\mathsf{EEP}(\mathcal{A})$, is to output the value of a given expression over $\mathcal{A}$. It can be viewed as a generalization of the well-known *word problem* $\mathsf{WP}(G)$ over a group, which asks whether a given product of group elements evaluate to the group identity.

In his seminal paper [27], Ladner proved that the circuit evaluation problem for the Boolean semiring $\mathbb{B}_2 = (\{0,1\}, \vee, \wedge)$ is P-complete. This result marks a cornerstone in the theory of P-completeness [22], and motivated the investigation of circuit evaluation problems for other algebraic structures. A large part of the literature is focused on arithmetic (semi)rings like $(\mathbb{Z}, +, \cdot)$, $(\mathbb{N}, +, \cdot)$

Authors' address: Moses Ganardi, ganardi@eti.uni-siegen.de; Danny Hucke, hucke@eti.uni-siegen.de; Daniel König, koenig@eti.uni-siegen.de; Markus Lohrey, lohrey@eti.uni-siegen.de, University of Siegen, Lehrstuhl für theoretische Informatik, Germany.

1942-3454/2018/1-ART1 $15.00
https://doi.org/0000001.0000001

or the max-plus semiring $(\mathbb{Z} \cup \{-\infty\}, \max, +)$ [2, 3, 26, 30, 31, 38]. These papers mainly consider circuits of polynomial formal degree. For commutative semirings, circuits of polynomial formal degree can be restructured into equivalent (unbounded fan-in) circuits of polynomial size and logarithmic depth [38]. This result leads to NC-algorithms for evaluating polynomial degree circuits over commutative semirings [30, 31]. Over non-commutative semirings, circuits of polynomial formal degree do in general not allow a restructuring into circuits of logarithmic depth [26]. In [31] it was shown that also for finite non-commutative semirings circuit evaluation is in NC for circuits of polynomial formal degree. On the other hand, the authors are not aware of any NC-algorithms for evaluating general (exponential degree) circuits over semirings. The lack of such algorithms is probably due to Ladner's result, which seems to exclude any efficient parallel algorithms (unless P = NC).

It is known that evaluating expressions over every fixed finite algebra is in NC^1 [28]. This was first proved by Buss for the case of Boolean formulas [13], where he also proved that the problem is in fact NC^1-complete. Later this result was extended to arbitrary finite algebras [28]. In the context of semigroups, the expression evaluation problem is usually called the word problem and has turned out to be useful in the description of complexity classes inside of NC^1. In [7, 9], the following dichotomy result was shown for finite semigroups: If the finite semigroup is solvable (meaning that every subgroup is a solvable group), then the word problem is in ACC^0, otherwise the word problem is NC^1-complete. A similar result holds for the circuit evaluation problem [12]: For solvable semigroups circuit evaluation is in NC (in fact, in DET, which is the class of all problems that are AC^0-reducible to the computation of an integer determinant [16, 17]), otherwise circuit evaluation is P-complete.

1.1 Main results

In this paper, we extend the previously mentioned work of [7, 9, 12] from finite semigroups to finite semirings. Let us first discuss the case of circuit evaluation. On first sight, it seems again that Ladner's result excludes efficient parallel algorithms: It is not hard to show that if the finite semiring has an additive identity 0 and a multiplicative identity $1 \neq 0$ (where 0 is not necessarily absorbing with respect to multiplication), then circuit evaluation is P-complete, see Corollary 4.5. Therefore, we take the most general reasonable definition of semirings: A semiring is a structure $(R, +, \cdot)$, where $R_+ := (R, +)$ is a commutative semigroup, $R_\bullet := (R, \cdot)$ is a semigroup, and $\cdot$ distributes (on the left and right) over $+$. In particular, we neither require the existence of a 0 nor a 1. We call a semiring $\{0,1\}$-*free* if there exists no subsemiring in which an additive identity 0 and a multiplicative identity $1 \neq 0$ exist. The first main result of this paper states:

Theorem 1.1. *Let R be a finite semiring. If R is $\{0,1\}$-free, then* $\mathsf{CEP}(R)$ *is* AC^0*-Turing-reducible to* $\mathsf{CEP}(R_+)$ *and* $\mathsf{CEP}(R_\bullet)$*. Otherwise* $\mathsf{CEP}(R)$ *is* P*-complete.*

Together with the result for semigroups from [12] (and the fact that commutative semigroups are solvable) we can conclude from Theorem 1.1 that there are only two obstacles to efficient parallel circuit evaluation: non-solvability of the multiplicative structure and the existence of a zero and a one (different from the zero) in a subsemiring. More precisely, we obtain the following classification:

Corollary 1.2. *Let R be a finite semiring.*

- *If R is $\{0,1\}$-free and both R_+ and $R_\bullet$ are aperiodic, then* $\mathsf{CEP}(R)$ *is in* NL *(nondeterministic logspace).*
- *If R is $\{0,1\}$-free and $R_\bullet$ is solvable, then* $\mathsf{CEP}(R)$ *is in* DET.
- *If R is not $\{0,1\}$-free or $R_\bullet$ is not solvable, then* $\mathsf{CEP}(R)$ *is* P*-complete.*

The hard part of the proof of Theorem 1.1 is the case when the semiring is $\{0, 1\}$-free. We will proceed in three steps. In the first two steps we reduce the circuit evaluation problem for a finite semiring R to the evaluation of a so-called *type admitting* circuit. This is a circuit where every gate evaluates to an element of the form eaf, where e and f are multiplicative idempotents of R. Moreover, these idempotents e and f have to satisfy a certain compatibility condition that will be expressed by a so called type function. In the last step, we present a parallel evaluation algorithm for type admitting circuits. Only for this last step we need the assumption that the semiring is $\{0, 1\}$-free.

Our second main result concerns the expression evaluation problem for finite semirings. Here we have a dichotomy similar to that found for the circuit evaluation problem:

Theorem 1.3. *Let R be a finite semiring. If R is $\{0, 1\}$-free, then* $\mathsf{EEP}(R)$ *is* TC^0*-reducible to* $\mathsf{WP}(R_\bullet)$. *Otherwise* $\mathsf{EEP}(R)$ *is* NC^1*-complete.*

To prove Theorem 1.3 we use the same proof strategy as for Theorem 1.1. To carry out the same reduction steps on expressions, i.e., circuits which are trees, we use the fact that for the given semiring expression an equivalent circuit of logarithmic depth can be computed in TC^0, which was proven in [20]. Moreover, this logarithmic depth circuit is given in the so-called extended connection (EC) representation, which is a set of tuples (u, p, v) where u and v are circuit gates and p is a bit string that addresses a path from gate u to gate v. It turns out that our circuit manipulations can be carried out in TC^0 on the EC-representations of log-depth circuits. Theorem 1.3 together with the results from [7, 9] imply:

Corollary 1.4. *Let R be a finite semiring. If R is $\{0, 1\}$-free and $R_\bullet$ is solvable, then* $\mathsf{EEP}(R)$ *is in* TC^0. *Otherwise* $\mathsf{EEP}(R)$ *is* NC^1*-complete.*

1.2 Application: Intersection non-emptiness problems

In Section 6 we present an application of our results for formal language theory. We consider the intersection non-emptiness problem for a given context-free language and a fixed regular language L. If the context-free language is given by an arbitrary context-free grammar, then we show that the intersection non-emptiness problem is P-complete as long as L is not empty (Lemma 6.1). It turns out that the reason for this are *non-productive* nonterminals, i.e., nonterminals which derive no word over the terminal alphabet. We therefore consider a restricted version of the intersection non-emptiness problem, where every nonterminal of the input context-free grammar must be productive. To avoid a promise problem (testing productivity of a nonterminal is P-complete), we in addition provide a witness of productivity for every nonterminal. This witness is an acyclic context-free grammar $\mathcal{H}$ whose productions are a subset of the productions of the input grammar $\mathcal{G}$ such that each nonterminal occurs on exactly one left-hand side of a production in $\mathcal{H}$. This ensures that $\mathcal{H}$ derives for every nonterminal A exactly one string that is a witness of the productivity of A. We then show that this restricted version of the intersection non-emptiness problem with the fixed regular language L is equivalent (with respect to constant depth reductions) to the circuit evaluation problem for a certain finite semiring that is derived from the regular language L.

As an application of the results for the expression evaluation problem, we consider the intersection non-emptiness problem for a given regular expression and a fixed regular language L. Again, we pose the restriction that the regular expression does not contain the empty set as an atomic expression; otherwise the problem is NC^1-complete for all non-empty languages L. Here it is also important that the given regular expression is balanced first, which is possible in TC^0 by [20].

1.3 Further related work

We mentioned already the existing work on circuit evaluation for infinite semirings. The question whether a given circuit over a polynomial ring evaluates to the zero polynomial is also known as polynomial identity testing. For polynomial rings over $\mathbb{Z}$ or $\mathbb{Z}_n$ ($n \geq 2$), polynomial identity testing has a co-randomized polynomial time algorithm [1, 23]. Moreover, the question, whether a deterministic polynomial time algorithm exists is tightly related to lower bounds in complexity theory, see [35] for a survey.

For finite non-associative groupoids, the complexity of circuit evaluation was studied in [32], and some of the results from [12] for semigroups were generalized to the non-associative setting. In [10], the problem of evaluating tensor circuits is studied. The complexity of this problem is quite high: Whether a given tensor circuit over the Boolean semiring evaluates to the (1×1)-matrix (0) is complete for nondeterministic exponential time. Finally, let us mention the papers [29, 37], where circuit evaluation problems are studied for the power set structures $(2^{\mathbb{N}}, +, \cdot, \cup, \cap, \overline{})$ and $(2^{\mathbb{Z}}, +, \cdot, \cup, \cap, \overline{})$, where $+$ and $\cdot$ are evaluated on sets via $A \circ B = \{a \circ b \mid a \in A, b \in B\}$.

A variant of our intersection non-emptiness problem was studied in [34]. There, a context-free language L is fixed, a (deterministic or non-deterministic) finite automaton $\mathcal{A}$ is the input, and the question is, whether $L \cap L(\mathcal{A}) = \emptyset$ holds. The authors present large classes of context-free languages such that for each member the intersection non-emptiness problem with a given regular language (specified by a non-deterministic automaton) is P-complete (resp., NL-complete).

2 PRELIMINARIES

Graphs in this paper will be finite, directed and node-labelled. Formally, a graph is a triple $\mathcal{G} = (V, E, \lambda)$ consisting of a finite set of nodes $V = V(\mathcal{G})$, a set of edges $E \subseteq V \times V$, and a labelling function $\lambda\colon V \to A$ into a set of labels A (for unlabelled graphs choose a singleton A). In an *ordered graph* $\mathcal{G}$ the outgoing edges of a node are linearly ordered. Formally, the edge relation E is a finite subset $E \subseteq V \times \mathbb{N} \times V$ such that, if $v \in V$ has k outgoing edges then for each $1 \leq i \leq k$ there exists a unique node v_i with $(v, i, v_i) \in E$. The number k is also called the *out-degree* of v. If each node has out-degree at most k, then $\mathcal{G}$ is called *k-ordered.* An (ordered) acyclic graph is called *(ordered) dag*. A *tree* $\mathcal{T}$ with root r is a graph such that for every node $v \in V(\mathcal{T})$ there exists exactly one path from r to v in $\mathcal{T}$. For a k-ordered graph $\mathcal{G}$, we define the set $\mathrm{path}(\mathcal{G})$ as the set of all triples $(u, \rho, v) \in V(\mathcal{G}) \times \{1, \ldots, k\}^* \times V(\mathcal{G})$ such that ρ is the sequence of edge labels along a path from node u to node v.

2.1 Algebraic and Boolean circuits

In this paper, an *(algebraic) structure* $\mathcal{A} = (A, f_1, \ldots, f_k)$ consists of a non-empty *domain* A and finitely many operations $f_i\colon A^{n_i} \to A$, where $n_i \in \mathbb{N} \cup \{*\}$ is the *arity* of f_i for $1 \leq i \leq k$. Arity $n_i = *$ means that the operation $f_i\colon A^* \to A$ gets an arbitrary number of input values from A (like for instance the Boolean OR- and AND-functions). We often identify the domain with the structure, if it is clear from the context.

A *circuit* over $\mathcal{A}$ with inputs $x_1, \ldots, x_n$ is a tuple $C = (V, E, \lambda, v_0)$ with the following properties:

- (V, E, λ) is an ordered dag, whose nodes are called *gates* and whose edges are called *wires.*
- $v_0 \in V$ is a distinguished *output gate.*
- $\lambda\colon V \to A \cup \{x_1, \ldots, x_n\} \cup \{f_1, \ldots, f_k\}$ is the node labelling function.

Moreover, the out-degree of a gate[1] v has to match the arity of its label: if $\lambda(v) \in A \cup \{x_1, \ldots, x_n\}$, then v must have out-degree 0 and is called an *input gate* of C. We also write $v \to \lambda(v)$ in this case.

[1]The out-degree of a gate is also called its fan-in in the literature. But since it is more convenient for our definition to direct edges towards the input gates, we prefer to use the term "out-degree".

If $\lambda(v)$ is a function f_i, then the out-degree of v is the arity of f_i, where the out-degree of v can be any number in case the arity of f_i is $*$. Moreover, v is called an *inner gate* of C in this case. If (v, i, v_i) for $1 \leq i \leq m$ are all the outgoing edges of v, then the gates $v_1, \ldots, v_m$ are the *input gates* of v. For simplicity we also write $v \to f_i(v_1, \ldots, v_m)$ in this case. For gates u, v we write $u \leq_C v$ if there exists a path from v to u in the circuit C. Thus $(V, \leq_C)$ is a partial order, whose minimal elements are the input gates. A circuit $C = (V, E, \lambda, v_0)$ is a *tree* if the graph (V, E) is a tree.

Fix a circuit $C = (V, E, \lambda, v_0)$ over $\mathcal{A}$. Every gate $v \in V$ computes a mapping $[v]\colon A^n \to A$ in the natural way. Intuitively, $[v](a_1, \ldots, a_n)$ is the value computed by gate v if the value a_i is assigned to the input x_i. Formally, let $\bar{a} = (a_1, \ldots, a_n)$ and define $[v](\bar{a})$ inductively as follows: If $v \to a \in A$, then $[v](\bar{a}) = a$. If $v \to x_i$, then $[v](\bar{a}) = a_i$. If $v \to f_i(v_1, \ldots, v_m)$ then $[v](\bar{a}) = f_i([v_1](\bar{a}), \ldots, [v_m](\bar{a}))$. If we want to make the underlying circuit C clear, we also write $[v]_C$ instead of $[v]$. The function computed by C is defined as $[C] = [v_0]$. If $n = 0$, then C is called *variable-free*. In this case, $[v] \in A$ is called the *value computed by gate* v. Moreover, $[C]$ is called the value computed by C.

A *Boolean circuit* is a circuit over a structure with domain $\{0, 1\}$. Specifically, we will consider such structures containing some of the following operations, where $p \geq 2$:

- $\textsc{not}\colon \{0,1\} \to \{0,1\}$ (Boolean negation),
- $\textsc{or}\colon \{0,1\}^* \to \{0,1\}$ (the Boolean or-function)
- $\textsc{and}\colon \{0,1\}^* \to \{0,1\}$ (the Boolean and-function),
- $\textsc{mod}_p\colon \{0,1\}^* \to \{0,1\}$ with $\textsc{mod}_p(a_1, \ldots, a_k) = 1$ iff p divides $\sum_{i=1}^k a_i$,
- $\textsc{maj}\colon \{0,1\}^* \to \{0,1\}$ with $\textsc{maj}(a_1, \ldots, a_k) = 1$ iff $\sum_{i=1}^k a_i \geq k/2$.

2.2 Computational complexity

For background in complexity theory we refer the reader to [5]. We assume that the reader is familiar with the complexity classes NL (non-deterministic logspace) and P (deterministic polynomial time). The canonical NL-complete problem is the graph accessibility problem GAP. A function is logspace-computable if it can be computed by a deterministic Turing-machine with a logspace-bounded work tape, a read-only input tape, and a write-only output tape. Note that the logarithmic space bound only applies to the work tape.

We use standard definitions concerning circuit complexity, see e.g. [40]. Consider a family of Boolean circuits $(C_n)_{n \geq 0}$, where C_n has n inputs $x_1, \ldots, x_n$. Such a family defines the language $L \subseteq \{0,1\}^*$ of all words $w \in \{0,1\}^*$ such that $[C_{|w|}](w) = 1$. The size (resp., depth) of the family $(C_n)_{n \geq 0}$ is the function mapping $n \in \mathbb{N}$ to the number of gates (resp., the length of a longest path from the output gate to an input gate) of the circuit C_n. The out-degree of the circuit family is the maximal out-degree of a gate in any circuit C_n; if this maximum does not exist, the circuit family has *unbounded out-degree*.

We only consider circuit families $(C_n)_{n \geq 0}$ of polynomially bounded size. For such a family, gates of C_n as well as the labels of gates can be encoded by bit strings of length $O(\log n)$. Moreover, we will only consider uniform circuit families $(C_n)_{n \geq 0}$, which roughly speaking means that the circuits C_n have to follow a common pattern. The precise notion of uniformity we are using is known as DLOGTIME-uniformity and is nowadays the standard choice of uniformity for low circuit complexity classes. To define DLOGTIME-uniformity one defines two representations for a circuit family $(C_n)_{n \geq 0}$:

- The *direct connection language* of $(C_n)_{n \geq 0}$ contains all tuples (n, u, i, v) and (n, u, t), where n is a binary encoded natural number, u and v are binary encoded gates of C_n, (u, i, v) is an edge of C_n, and t is the (binary encoding of the) label of gate u.

- The *extended connection language* of $(C_n)_{n\geq 0}$, which is only defined if the circuit family has out-degree two, contains all tuples (n, u, ρ, v, t), where n is a binary encoded natural number, u and v are binary encoded gates of C_n, $\rho \in \{1,2\}^*$ has length at most $\log_2 n$, $(u, \rho, v) \in \mathsf{path}(C)$ and t is the (binary encoding of the) label of gate u.

A circuit family $(C_n)_{n\geq 0}$ of out-degree two is U_E-uniform if the extended connection language can be decided in linear time on a random access Turing machine, which means that the running time is in $O(\log n)$ for an input tuple (n, u, ρ, v, t) (since n, u, ρ, v, t are encoded by bit strings of length $O(\log n)$). A circuit family $(C_n)_{n\geq 0}$ (of possibly unbounded out-degree) is U_D-uniform if the directed connection language can be decided in linear time. These notions both strengthen logspace uniformity, where it is required that the mapping $a^n \mapsto C_n$ is logspace computable.

The following two standard circuit complexity classes will be used in this paper:

- AC^0: the class of all languages that can be recognized by a U_D-uniform circuit family of polynomial size and constant depth built up from the Boolean functions NOT, OR, and AND.
- NC^1: the class of all languages that can be recognized by a U_E-uniform circuit family of out-degree two, polynomial size and depth $O(\log n)$ built up from the Boolean functions NOT, OR, and AND.
- NC^i for $i \geq 2$: the class of all languages that can be recognized by a logspace uniform circuit family of out-degree two, polynomial size and depth $O(\log^i n)$ built up from the Boolean functions NOT, OR, and AND.
- $\mathsf{NC} = \bigcup_{i\geq 1} \mathsf{NC}^i$.

In order to compute a function $f\colon \{0,1\}^* \to \{0,1\}^*$ with a circuit family, we encode f by the language $L_f = \{1^i 0 w \mid w \in \{0,1\}^*, \text{ the } i\text{-th bit of } f(w) \text{ is } 1\}$. We only consider functions $f\colon \{0,1\}^* \to \{0,1\}^*$ such that $|f(w)|$ is polynomially bounded by $|w|$.

Unless noted otherwise, hardness results in this paper refer to AC^0*-many-one reductions*, i.e., many-one reductions that can be computed by AC^0 circuit families. We also use the standard notion of AC^0-Turing-reducibility: A function $f\colon \{0,1\}^* \to \{0,1\}$ is AC^0*-Turing-reducible* to functions $f_1, \dots, f_k\colon \{0,1\}^* \to \{0,1\}$ if f can be computed with a U_D-uniform circuit family of polynomial size and constant depth built up from the Boolean functions NOT, OR, AND, and $f_1, \dots, f_k$. The class $\mathsf{AC}^0(f_1, \dots, f_k)$ contains all functions that are AC^0-Turing-reducible to $f_1, \dots, f_k$. By taking the characteristic function of a language, we can also allow a language $L_i \subseteq \{0,1\}^*$ in place of f_i. Similarly, by taking the (characteristic function of the) language L_{f_i}, we can also allow functions $f_i\colon \{0,1\}^* \to \{0,1\}^*$. The two classes $\mathsf{ACC}^0 := \bigcup_{p\geq 2} \mathsf{AC}^0(\textsc{mod}_p)$ and $\mathsf{TC}^0 := \mathsf{AC}^0(\textsc{maj})$ are well-studied in circuit complexity.

Let $\mathsf{DET} = \mathsf{AC}^0(\textsc{det})$, where DET is the function that maps a binary encoded integer matrix to the binary encoding of its determinant, see [16]. Actually, Cook defined DET as $\mathsf{NC}^1(\textsc{det})$ [16], but the above definition via AC^0-circuits seems to be more natural. For instance, it implies that DET is equal to the #L-hierarchy, see also the discussion in [17]. We defined DET as a function class, but the definition can be extended to languages by considering their characteristic functions. The following inclusions are well known:

$$\mathsf{AC}^0 \subseteq \mathsf{ACC}^0 \subseteq \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{NL} \subseteq \mathsf{DET} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

2.3 Tree and graph representations

Recall the definition of an algebraic structure $\mathcal{A} = (A, f_1, \dots, f_k)$ and a circuit C over $\mathcal{A}$ from Section 2.1. For the rest of the paper, we only consider the case that all operations f_i are of type $f_i\colon A^{n_i} \to A$ with $n_i \in \mathbb{N}$. Operations of type $f_i\colon A^* \to A$ were only needed for the definition of circuit complexity classes, and the same holds for circuits that are not variable-free. Hence, we only consider variable-free circuits in the following. Let us fix an algebraic structure $\mathcal{A} = (A, f_1, \ldots, f_k)$ as above for the following definitions.

When dealing with low complexity classes, the precise input representation is quite often very important. The standard input representation of a circuit $C = (V, E, \lambda, v_0)$ is its *pointer representation*, where we assume w.l.o.g. that $V = \{1, \ldots, n\}$ for some n and $v_0 = 1$: The pointer representation stores a list of the node labels $\lambda(v)$ and a list of all edges. In order to ensure acyclicity of the circuit, we can assume that $i < j$ whenever there is an edge from node i to node j.

For trees (which were defined as particular circuits) the pointer representation is not suitable when working with complexity classes below logarithmic space, since deciding whether a node v is an ancestor of another node u is already hard for logarithmic space [19]. Hence, we will deal with two alternative representations of trees. The first one is the *ancestor representation*, which consists of the pointer representation of a tree $\mathcal{T}$ together with (a list of all pairs from) the ancestor relation $\leq_{\mathcal{T}}$. In order to emphasize that a tree $\mathcal{T}$ is given in ancestor relation, we also write the tree as $(\mathcal{T}, \leq_{\mathcal{T}})$.

It is known that the ancestor representation of a tree is equivalent (with respect to TC^0-reductions) to its term representation. The set of *terms* (or *expressions*) over the structure $\mathcal{A}$ is the set of all strings over the alphabet $A \cup \{f_1, \ldots, f_k\}$ that can be constructed inductively as follows: Every element of A is a term over $\mathcal{A}$ and if $t_1, \ldots, t_{n_i}$ are terms over $\mathcal{A}$ then also $f_i t_1 \cdots t_{n_i}$ is a term over $\mathcal{A}$. The following result seems to be folklore but we provide a proof because we were not able to find a reference.

Proposition 2.1. *Let $\mathcal{A} = (\{a\}, f)$ where f is a binary symbol. Deciding whether a string over the alphabet $\{f, a\}$ is a valid term over $\mathcal{A}$ is TC^0-hard with respect to AC^0-Turing-reductions.*

Proof. We reduce from the TC^0-complete equality problem [15], i.e., the problem whether a given word $w \in \{f, a\}^*$, contains an equal number of f's and a's. This is the case if and only if $f^n\, w\, a^{n+1}$ is a valid term, where $n = |w|$. □

The value $[t] \in A$ of a term over $\mathcal{A}$ is defined in the natural way. A tree $\mathcal{T} = (V, E, \lambda, v_0)$ defines a term in the natural way. Formally, we assign to each gate $v \in V$ a term t_v as follows: If v is an input gate then $t_v = \lambda(v)$ and if $v \to f(v_1, \ldots, v_n)$ then $t_v = f t_{v_1} \cdots t_{v_n}$. We call the term t_{v_0} the term representation of the tree $\mathcal{T}$. In fact, this construction can be done for an arbitrary circuit $C = (V, E, \lambda, v_0)$. The term t_{v_0} is then called the unfolding of the circuit. Of course, it evaluates to $[C]$.

Convention. *Unless otherwise specified,*

- *circuits are always represented in pointer representation and*
- *trees are always represented in ancestor representation.*

Finally, we introduce the extended connection representation of a k-ordered graph, which is very similar to the extended connection language of a circuit family defined in Section 2.2. In fact, we will only need the extended connection representation for circuits. The *extended connection representation*, briefly *EC-representation*, of a k-ordered graph $\mathcal{G} = (V, E, \lambda)$, denoted by $\mathrm{ec}(\mathcal{G})$, is the pair $(\mathcal{G}, P)$ where P is the set of all tuples $(u, \rho, v) \in \mathrm{path}(\mathcal{G})$ such that $|\rho| \leq \log_n |V|$. The EC-representation of a circuit will be only useful for circuits of logarithmic depth.

The relationships between the different representations are summarized in the following lemma.

Lemma 2.2 ([18, 20]). *The following holds:*

(1) *There is a TC^0-computable function converting the term representation of a tree into its ancestor representation, and vice versa.*

(2) *There is a* TC^0*-computable function converting the ancestor representation of a tree into its EC-representation.*

(3) *For every fixed* $c > 0$ *there is a* TC^0*-computable function which, given a circuit* C *in EC-representation of depth* $\leq c \cdot \log_2 |C|$*, outputs the unfolding of* C *(in term or ancestor representation).*

2.4 Preliminary algorithmic results for circuits and trees

It is sometimes convenient to compute arbitrary terms in circuit gates, namely terms built from constants, gates and operations of the structure. For instance, over a semiring $(R, +, \cdot)$ we might consider the following circuit: $v_0 \to a \cdot v_1 \cdot b + v_2$, $v_1 \to v_3$, $v_2 \to v_3 \cdot v_3$, $v_3 \to c$. Formally, this is a circuit over an algebraic structure extended by certain *term functions*; in our example this is structure $(R, +, \cdot, f, g, i)$ where $f(x, y) = a \cdot x \cdot b + y$, $g(x) = x \cdot x$ and i is the identity function. The gate v_1 (with $v_1 \to v_3$) is called a *copy gate*. Allowing copy gates is often convenient for the construction of circuits. We say that a circuit is in *normal form* to emphasize that the circuit only contains the operators of the initial algebraic structure. By the following lemma, we can always assume that circuits are in normal form:

Lemma 2.3 ([25, Lemma 1]). *A given circuit can be transformed in logspace into an equivalent normal form circuit.*

Also for trees we can allow arbitrary terms as node labels, with the restriction that every non-output gate of the tree has exactly one occurrence among the terms that appear as node labels. For instance, $v_0 \to a \cdot v_1 \cdot b + v_2$, $v_1 \to v_3$, $v_2 \to v_4 \cdot v_5$, $v_3 \to c$, $v_4 \to d$, $v_5 \to a$, where a, b, c, d are constants, specifies a valid tree. The ancestor relation is the reflexive and transitive closure of $\{(v_0, v_1), (v_0, v_2), (v_1, v_3), (v_2, v_4), (v_2, v_5)\}$.

Lemma 2.4. *A given tree* $(\mathcal{T}, \leq_{\mathcal{T}})$ *can be transformed in* TC^0 *into an equivalent normal form tree* $(\mathcal{T}', \leq_{\mathcal{T}'})$*.*

Proof. As for circuits (see [25, Lemma 1]) we can easily compute the intermediate gates in AC^0 for right-hand sides with multiple operators. Then we have to recompute the ancestor relation, which is possible by keeping track of which intermediate gates originate from which gates. Eliminating the copy gates is possible in TC^0 because we have access to the ancestor relation. □

A function I mapping ordered graphs (over a fixed set of node labels A) to ordered graphs (over a possibly different fixed set A' of node labels) is called a *guarded transduction* if there exist numbers $m, c \in \mathbb{N}$ such that every ordered graph $\mathcal{G} = (V, E, \lambda \colon V \to A)$ is mapped to an ordered graph $I(\mathcal{G}) = (V', E', \lambda' \colon V' \to A')$ with the following properties:

- $V' \subseteq V^m \times \{1, \ldots, c\}$, and
- for each edge $((u_1, \ldots, u_m, a), (v_1, \ldots, v_m, b)) \in E'$ and every $1 \leq j \leq m$ there exists $1 \leq i \leq m$ such that $(u_i, v_j) \in E$ or $u_i = v_j$.

We will need the following lemma:

Lemma 2.5 ([20]). *For every* TC^0*-computable guarded graph transduction* I *there exists a* TC^0*-computable function that maps* $\mathrm{ec}(\mathcal{G})$ *to* $\mathrm{ec}(I(\mathcal{G}))$ *for all* k*-ordered graphs* $\mathcal{G}$*.*

3 SEMIGROUPS AND SEMIRINGS

Let $\mathcal{A} = (A, f_1^{\mathcal{A}}, \ldots, f_k^{\mathcal{A}})$, $\mathcal{B} = (B, f_1^{\mathcal{B}}, \ldots, f_k^{\mathcal{B}})$ be two algebraic structures where $f_i^{\mathcal{A}}$ and $f_i^{\mathcal{B}}$ have the same arity for all $1 \leq i \leq k$. A *homomorphism* from $\mathcal{A}$ to $\mathcal{B}$ is a function $h \colon A \to B$ such that

$$h(f_i^{\mathcal{A}}(a_1, \ldots, a_{n_i})) = f_i^{\mathcal{B}}(h(a_1), \ldots, h(a_{n_i}))$$

for all $a_1, \ldots, a_n \in A$ and all operations $f_i \colon A^{n_i} \to A$. A bijective homomorphism is called an *isomorphism*.

We mainly deal with semigroups and semirings. In the following two subsections we present the necessary background. For further details on semigroup theory (resp., semiring theory) see [33] (resp., [21]).

3.1 Semigroups

A *semigroup* $(S, \cdot)$ is an algebraic structure with a single associative binary operation $\cdot \colon S \times S \to S$. We usually write st for $s \cdot t$. If $st = ts$ for all $s, t \in S$, then S is *commutative*. A subset $T \subseteq S$ is a *subsemigroup* of S if it is closed under $\cdot$. A subset $I \subseteq S$ is called a *semigroup ideal* if for all $s \in S$, $a \in I$ we have $sa, as \in I$. An element $e \in S$ is called *idempotent* if $e^2 = e$. It is well-known that for every finite semigroup S and $s \in S$ there exists an $n \geq 1$ such that s^n is idempotent. In particular, every finite semigroup contains an idempotent element. By taking the smallest common multiple of all these n, one obtains a number $\omega \geq 1$ such that s^ω is idempotent for all $s \in S$. The set of all idempotents of S is denoted with $E(S)$, or simply E. If S is finite, then $S^n = SES$ for all $n \geq |S|$. For a set Σ, the *free semigroup* generated by Σ is the set Σ^+ of all finite non-empty words over Σ together with the operation of concatenation.

A *monoid* $(M, \cdot, 1)$ is a semigroup $(M, \cdot)$ with an identity element $1 \in M$, i.e., $1m = m1 = m$ for all $m \in M$. Every semigroup S can be extended to a monoid $S^1 = S \cup \{1\}$ by adding a new identity element 1, i.e., we extend the multiplication to $S^1 = S \cup \{1\}$ by setting $1s = s1 = s$ for all $s \in S \cup \{1\}$. A subset $N \subseteq M$ is a *submonoid* of M if it is closed under $\cdot$ and has an identity element $e \in N$. Notice that we do not require that $1 = e$ but clearly e must be an idempotent element of M. In fact, for every semigroup S and every idempotent $e \in S$, the set $eSe = \{ese \mid s \in S\}$ is a submonoid of S with identity e, which is also called a *local submonoid* of S. The local submonoid eSe is the maximal submonoid of S whose identity element is e. If each local submonoid eSe is a group, then S is a *local group*. A semigroup S is *aperiodic* if every subgroup of S is trivial. A semigroup S is *solvable* if every subgroup G of S is a solvable group, i.e., for the series defined by $G_0 = G$ and $G_{i+1} = [G_i, G_i]$ (the commutator subgroup of G_i) there exists an $i \geq 0$ with $G_i = 1$. Since Abelian groups are solvable, every commutative semigroup is solvable.

3.2 Semirings

A *semiring* $(R, +, \cdot)$ consists of a non-empty set R with two binary operations $+$ and $\cdot$ such that $R_+ := (R, +)$ is a commutative semigroup, $R_\bullet := (R, \cdot)$ is a semigroup, and $\cdot$ left- and right-distributes over $+$, i.e., $a \cdot (b + c) = ab + ac$ and $(a + b) \cdot c = ab + ac$. Note that we neither require the existence of an additive identity 0 nor the existence of a multiplicative identity 1. We call $(R, +, \cdot, 0, 1)$ a $\{0,1\}$-*semiring* if $(R, +, \cdot)$ is a semiring, $(R, +, 0)$ and $(R, \cdot, 1)$ are monoids, and $0 \neq 1$. Note that in a $\{0,1\}$-semiring we do *not* require that 0 is absorbing, i.e., $a \cdot 0 = 0 \cdot a = 0$.

A subset $T \subseteq R$ is a *subsemiring* if it is closed under $+$ and $\cdot$. For a non-empty subset $T \subseteq R$ we denote by $\langle T \rangle$ the subsemiring generated by T, i.e., the smallest subsemiring of R containing T. For $n \geq 1$ and $r \in R$ we write $n \cdot r$ or just nr for $\sum_{i=1}^{n} r$. With $E(R) = \{e \in R \mid e^2 = e\}$ we denote the set of multiplicative idempotents of R. Examples for semirings are the *Boolean semiring* $\mathbb{B}_2 = (\{0,1\}, \vee, \wedge)$ or the ring $\mathbb{Z}_q = (\{0, \ldots, q-1\}, +, \cdot)$.

For a given non-empty set Σ, the *free semiring* $\mathbb{N}[\Sigma]$ generated by Σ consists of all mappings $\alpha \colon \Sigma^+ \to \mathbb{N}$ such that the support of α defined by $\mathrm{supp}(\alpha) := \{w \in \Sigma^+ \mid \alpha(w) \neq 0\}$ is finite and non-empty. Addition is defined pointwise, i.e., $(\alpha + \beta)(w) = \alpha(w) + \beta(w)$, and multiplication is defined by the convolution: $(\alpha \cdot \beta)(w) = \sum_{w=uv} \alpha(u) \cdot \beta(v)$, where the sum is taken over all factorizations $w = uv$ with $u, v \in \Sigma^+$. We view an element $\alpha \in \mathbb{N}[\Sigma]$ as a non-commutative polynomial

$\sum_{w \in \mathsf{supp}(\alpha)} \alpha(w) \cdot w$. Then addition (resp. multiplication) in $\mathbb{N}[\Sigma]$ corresponds to addition (resp. multiplication) of non-commutative polynomials. Words $w \in \mathsf{supp}(\alpha)$ are also called *monomials* of α. A word $w \in \Sigma^+$ is identified with the non-commutative polynomial $1 \cdot w$, i.e., the mapping α with $\mathsf{supp}(\alpha) = \{w\}$ and $\alpha(w) = 1$. For every semiring R which is generated by Σ there exists a canonical surjective homomorphism from $\mathbb{N}[\Sigma]$ to R which evaluates non-commutative polynomials over Σ. Since a semiring is not assumed to have a multiplicative identity (resp., additive identity), we have to exclude the empty word from $\mathsf{supp}(\alpha)$ for every $\alpha \in \mathbb{N}[\Sigma]$ (resp., exclude the mapping with empty support from $\mathbb{N}[\Sigma]$).

A crucial definition in this paper is that of a $\{0,1\}$-free semiring. A semiring R is $\{0,1\}$*-free* if it does *not* contain a subsemiring which is a $\{0,1\}$-semiring. The class of *finite* $\{0,1\}$-free semirings has several characterizations:

Lemma 3.1. *For a finite semiring R, the following are equivalent:*

(1) *R is not $\{0,1\}$-free.*
(2) *R contains $\mathbb{B}_2$ or the ring $\mathbb{Z}_q$ for some $q \geq 2$ as a subsemiring.*
(3) *R is divided by $\mathbb{B}_2$ or $\mathbb{Z}_q$ for some $q \geq 2$ (i.e., $\mathbb{B}_2$ or $\mathbb{Z}_q$ is a homomorphic image of a subsemiring of R).*
(4) *There exist elements $0, 1 \in R$ such that $0 \neq 1$, $0+0=0$, $0+1=1$, $0 \cdot 1 = 1 \cdot 0 = 0 \cdot 0 = 0$, and $1 \cdot 1 = 1$ (but $1+1 \neq 1$ is possible).*

Proof. The implication from (2) to (3) is of course trivial. It therefore suffices to show the implications (1) $\Rightarrow$ (2), (3) $\Rightarrow$ (4), and (4) $\Rightarrow$ (1).

(1) $\Rightarrow$ (2): Let $(T, +, \cdot, 0, 1)$ be a $\{0,1\}$-subsemiring of R. Note that $0 \cdot 0 = 0 \cdot 0 + 0 = 0 \cdot 0 + 1 \cdot 0 = (0+1) \cdot 0 = 1 \cdot 0 = 0$. Let $T' = \{0\} \cup \{k \cdot 1 \mid k \in \mathbb{N}\}$, which is the subsemiring generated by these elements. It is isomorphic to some semiring $B(t,q)$ ($t \geq 0$, $q \geq 1$), which is the semiring $(\mathbb{N}, +, \cdot)$ modulo the congruence relation $\theta_{t,q}$ defined by

$$i\,\theta_{t,q}\,j \iff i = j \text{ or } [i,j \geq t \text{ and } i \equiv j \pmod{q}].$$

Since $0 \neq 1$, we have $(t,q) \neq (0,1)$. If $t = 0$, then $B(0,q)$ is isomorphic to $\mathbb{Z}_q$ for $q \geq 2$. If $t \geq 1$, then choose $a \geq t$ such that q divides a, for example $a = qt$. Then $\{0, a \cdot 1\}$ is a subsemiring isomorphic to the Boolean semiring $\mathbb{B}_2$.

(3) $\Rightarrow$ (4): Assume that $\varphi \colon T \to T'$ is a surjective homomorphism from a subsemiring T of R to T', where T' is $\mathbb{B}_2$ or $\mathbb{Z}_q$ with $q \geq 2$. In particular, there exist $0, 1 \in T'$ with $0 \neq 1$, $0+0=0$, $0+1=1$, $0 \cdot 1 = 1 \cdot 0 = 0 \cdot 0 = 0$, and $1 \cdot 1 = 1$. Let $n \geq 1$ be such that $n \cdot x$ is additively idempotent and x^n is multiplicatively idempotent for all $x \in R$. Then $n \cdot x^n$ is additively and multiplicatively idempotent for all $x \in R$. Let $a, e \in T$ be such that $\varphi(a) = 0$ and $\varphi(e) = 1$. Since $\varphi(n \cdot a^n) = 0$ and $\varphi(e^n) = 1$, we can replace a by $n \cdot a^n$ and e by e^n. Then, $a + a = aa = a$ and $ee = e$. For $a' = n \cdot (eae)^n$ we have $\varphi(a') = 0$ and $a'e = ea' = a' + a' = a'a' = a'$. For $e' = a' + e$ we have $\varphi(e') = 1$ (hence, $a' \neq e'$) and $e'e' = a'a' + a'e + ea' + ee = a' + e = e'$, $a' + e' = a' + a' + e = e'$. Furthermore, we have $a'e' = a'(a'+e) = a' + a'e = a'$ and similarly $e'a' = a'$. Hence, a' and e' satisfy all equations from point 4.

(4) $\Rightarrow$ (1): Assume that there exist elements $0, 1 \in R$ such that $0 \neq 1$, $0+0=0$, $0+1=1$, $0 \cdot 1 = 1 \cdot 0 = 0 \cdot 0 = 0$, and $1 \cdot 1 = 1$. Consider the subsemiring generated by $\{0,1\}$, which is $\{0\} \cup \{n \cdot 1 \mid n \geq 1\}$. By the above identities 0 (resp., 1) is an additive (resp., multiplicative) identity in this subsemiring. □

As a consequence of Lemma 3.1 (point 4), one can check in time $O(n^2)$ for a semiring of size n whether it is $\{0,1\}$-free. We will not need this fact, since in our setting the semiring will be always fixed, i.e., not part of the input. Moreover, the class of all $\{0,1\}$-free semirings is closed under taking subsemirings (this is trivial) and taking homomorphic images (by point 3). Finally, the class of $\{0,1\}$-free semirings is also closed under direct products. To see this, assume that $R \times R'$ is not $\{0,1\}$-free. Hence, there exists a subsemiring T of $R \times R'$ with an additive zero $(0, 0')$ and a multiplicative one $(1, 1') \neq (0, 0')$. W.l.o.g. assume that $0 \neq 1$. Then the projection $\pi_1(T)$ onto the first component is a subsemiring of R, where 0 is an additive identity and $1 \neq 0$ is a multiplicative identity. By these remarks, the class of $\{0,1\}$-free finite semirings forms a *pseudo-variety* of finite semirings. Again, this fact will not be used in the rest of the paper, but it might be of independent interest.

4 EVALUATING CIRCUITS AND EXPRESSIONS

4.1 Evaluation problems

Let $\mathcal{A} = (A, f_1, \ldots, f_n)$ be an algebraic structure with a countable domain A. We assume some fixed finite representation of elements from A. The *circuit evaluation problem* $\mathsf{CEP}(\mathcal{A})$ is the following decision problem:

Input A normal form circuit C over $\mathcal{A}$ and an element $a \in A$.
Question Does $[C] = a$ hold?

The *word problem* $\mathsf{WP}(S)$ of a semigroup S is defined as follows:

Input A word $s_1 \cdots s_n \in S^+$ and an element $s \in S$.
Question Does $s_1 \cdot \ldots \cdot s_n = s$ hold in S?

The natural generalization of the word problem to nonassociative structures is the expression evaluation problem $\mathsf{EEP}(\mathcal{A})$ for $\mathcal{A}$:

Input An expression t over $\mathcal{A}$ and an element $a \in A$.
Question Does $[t] = a$ hold?

For a semigroup S one can clearly reduce $\mathsf{EEP}(S)$ to $\mathsf{WP}(S)$ in TC^0. As explained in Section 2.3 we can convert in TC^0 an expression into an equivalent tree $\mathcal{T}$ in ancestor representation and vice versa. Therefore from Section 5 on, we consider the equivalent *tree evaluation problem* $\mathsf{TEP}(\mathcal{A})$, which is technically more convenient.

Input A normal form tree $(\mathcal{T}, \leq_{\mathcal{T}})$ over $\mathcal{A}$ and an element $a \in A$.
Question Does $[\mathcal{T}] = a$ hold?

We emphasize that the ancestor relation $\leq_{\mathcal{T}}$ is part of the input. Also note that in the problem $\mathsf{CEP}(\mathcal{A})$ (resp., $\mathsf{TEP}(\mathcal{A})$), we always assume that the input circuit (resp., tree) is in normal form.

4.2 Evaluation over finite structures, semigroups and semirings

As mentioned in the introduction, evaluation problems for finite structures, in particular semigroups, have already been thoroughly studied. Clearly, for every finite structure the circuit evaluation problem can be solved in polynomial time by evaluating all gates along the partial order $\leq_C$. Furthermore, it is known that expressions over any finite structure can be evaluated in NC^1 [28].

Theorem 4.1 ([28]). *For every finite algebraic structure $\mathcal{A}$, the problem* $\mathsf{CEP}(\mathcal{A})$ *is in* P *and the problem* $\mathsf{EEP}(\mathcal{A})$ *is in* NC^1.

For finite semigroups the complexity of the circuit evaluation problem and the word problem is very well-understood. We briefly summarize the results in the following:

Theorem 4.2. *Let S be a finite semigroup.*

(1) *If S is aperiodic, then* $\mathsf{CEP}(S)$ *is in* NL *[12] and* $\mathsf{WP}(S)$ *is in* AC^0 *[14].*
(2) *If S is not a local group, then* $\mathsf{CEP}(S)$ *is* NL*-hard [12].*
(3) *If S is solvable, then* $\mathsf{CEP}(S)$ *is in* DET *[12] and* $\mathsf{WP}(S)$ *is in* ACC^0 *[7].*
(4) *If S is not solvable, then* $\mathsf{CEP}(S)$ *is* P*-complete [12] and* $\mathsf{WP}(S)$ *is* NC^1*-complete [7].*

Some remarks should be made:

- In [12], the results for the circuit evaluation problem from (1) and (3) are only shown for monoids, but the extension to semigroups is straightforward, by going from a semigroup S over to the monoid S^1. The subgroups of S^1 are exactly the subgroups of S together with $\{1\}$, and hence S is solvable (resp., aperiodic) if and only if S^1 is solvable (resp., aperiodic).
- In [12], statement (2) is only shown for the case that S contains a non-trivial aperiodic monoid. But if S contains a local submonoid eSe which is not a group, then eSe contains an idempotent element f which is different from e. Then $\{e, f\}$ is a non-trivial aperiodic monoid.
- In [12], the authors use the original definition $\mathsf{DET} = \mathsf{NC}^1(\det)$ of Cook. But the arguments in [12] actually show that for a finite solvable semigroup, $\mathsf{CEP}(S)$ belongs to $\mathsf{AC}^0(\det)$ (which is our definition of DET).
- In [12], the authors study two versions of the circuit evaluation problem for a semigroup S: What we call $\mathsf{CEP}(S)$ is called $\mathsf{UCEP}(S)$ (for "unrestricted circuit evaluation problem") in [12]. The problem $\mathsf{CEP}(S)$ is defined in [12] as the circuit evaluation problem, where in addition the input circuit must have the property that the output gate has no ingoing edges and all gates are reachable from the output gate. Since graph reachability is in NL, the difference between the two variants is only relevant for classes below NL. We only consider the unrestricted version of the circuit evaluation problem (where the input circuit is arbitrary). To keep notation simple, we decided to refer with CEP to the unrestricted version.

For finite semirings much less is known about the circuit and the expression evaluation problem. The classical results by Ladner and Buss for the Boolean circuit value problem and the Boolean formula value problem, respectively, can be stated as follows:

Theorem 4.3 ([13, 27]). *For the Boolean semiring $\mathbb{B}_2 = (\{0,1\}, \vee, \wedge)$, the problem* $\mathsf{CEP}(\mathbb{B}_2)$ *is* P*-complete and the problem* $\mathsf{EEP}(\mathbb{B}_2)$ *is* NC^1*-complete.*

Proposition 4.4. *For any $q \geq 2$ the problem* $\mathsf{CEP}(\mathbb{Z}_q)$ *is* P*-complete and the problem* $\mathsf{EEP}(\mathbb{Z}_q)$ *is* NC^1*-complete.*

Proof. One can simulate conjunction and negation over $\{0,1\}$ in $\mathbb{Z}_q$ for $q \geq 2$: A conjunction $z \to x \wedge y$ is simulated by $z \to x \cdot y$ and a negation $y \to \neg x$ is simulated by $y \to 1 - x$, which is formally $1 + (q-1) \cdot x$. Hence, a disjunction $z \to x \vee y$ can be simulated by $z \to 1 - (1-x) \cdot (1-y)$. This proves that $\mathsf{CEP}(\mathbb{Z}_q)$ is P-hard (under AC^0-many-one-reductions), and that $\mathsf{EEP}(\mathbb{Z}_q)$ is NC^1-hard under TC^0-many-one-reductions, since one can convert to the tree representation first and carry out the simulation by a guarded transduction.

In fact, $\mathsf{EEP}(\mathbb{Z}_q)$ is NC^1-hard under AC^0-many-one reductions, which we prove by a reduction from the word problem of the symmetric group S_m for some $m \geq 5$. Since S_m is nonsolvable for all $m \geq 5$, the word problem $\mathsf{WP}(S_m)$ is NC^1-complete under AC^0-many-one reductions [7, 8].[2] We view S_m as the subgroup of permutation matrices in $\mathbb{Z}_q^{m \times m}$, i.e., matrices which contain exactly one 1-entry in each row and each column, and otherwise only 0-entries. Given permutation matrices $M_1, \ldots, M_n \in \mathbb{Z}_q^{m \times m}$ it suffices to construct for each $1 \leq i, j \leq m$ an expression $t_{i,j}$ over $\mathbb{Z}_q$

[2]Barrington shows in [7] only completeness with respect to AC^0-Turing-reductions. Barrington, Immerman and Straubing [8] prove completeness with respect to AC^0-many-one reductions (in fact, $\mathsf{DLOGTIME}$-reductions).

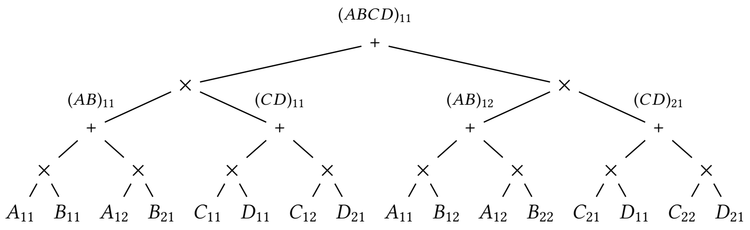
$(ABCD)_{11}$
$(AB)_{11}$
$(CD)_{11}$
$(AB)_{12}$
$(CD)_{21}$
A_{11} B_{11} A_{12} B_{21} C_{11} D_{11} C_{12} D_{21} A_{11} B_{12} A_{12} B_{22} C_{21} D_{11} C_{22} D_{21}

Fig. 1. An expression computing the $(1,1)$-entry of the (2×2)-matrix product $A \cdot B \cdot C \cdot D$.

computing the (i,j)-entry of the matrix product $M_1 \cdots M_n$. The final expression is $t = \prod_{i=1}^{m} t_{i,i}$. Since $M_1, \ldots, M_n$ are permutation matrices, t evaluates to 1 if and only if $M_1 \cdots M_n$ is the identity matrix.

For simplification we assume that the length n and the dimension $m \geq 5$ are powers of 2. The product $M_1 \cdots M_n$ can be computed by a full binary tree of height $\log n$ and hence each entry can be computed by a full binary tree of height $(\log_2 m + 1) \cdot \log_2 n = O(\log n)$ over the ring $\mathbb{Z}_q$, as illustrated in Figure 1. We argue that the term representation of this tree can be computed in AC^0.

Suppose we want to compute the symbol of the expression at a given position s. First we compute the word $\rho \in \{0,1\}^*$ of length $O(\log n)$ which describes the path from the root to the node at position s, which is possible in AC^0 due to the regular structure of the tree. Notice that the node addressed by ρ is a leaf if and only if $|\rho| = (\log m + 1) \cdot \log n$. If $|\rho| < (\log m + 1) \cdot \log n$, then the node is a $\times$-gate if and only if $|\rho| + 1$ is divisible by $\log m + 1$.

Now let ρ be a string of length $(\log m + 1) \cdot \log n$, which addresses a leaf node labelled by the (i,j)-entry of matrix M_k. Projecting ρ to all positions divisible by $(\log m + 1)$ yields the binary representation of the number k. The position (i,j) can be easily calculated by a finite automaton which reads ρ. This automaton can be simulated in AC^0 by guessing and verifying a run of length $O(\log n)$. $\square$

By Lemma 3.1, any finite semiring R which is not $\{0,1\}$-free contains either $\mathbb{B}_2$ or $\mathbb{Z}_q$ for some $q \geq 2$, which implies the lower bounds in Theorem 1.1 and Theorem 1.3:

Corollary 4.5. *If a finite semiring R is not $\{0,1\}$-free, then* $\mathsf{CEP}(R)$ *is* P*-complete and* $\mathsf{EEP}(R)$ *is* NC^1*-complete.*

Let us finally state the following result from [20] which is needed for the proof of Theorem 1.3:

Proposition 4.6. *There is a* TC^0*-computable function which transforms a given tree $(\mathcal{T}, \leq_{\mathcal{T}})$ over a semiring into an equivalent tree $(\mathcal{T}', \leq_{\mathcal{T}'})$ in normal form of depth $O(\log n)$, where n is the number of nodes of $\mathcal{T}$.*

Note that the normal form of the output tree means that the output tree is a binary tree of depth $O(\log n)$.

4.3 Circuits over power semirings

Let us present an application of Corollary 1.2 and Corollary 1.4. An important semigroup construction found in the literature is the *power semiring*. For a semigroup S one defines $\mathcal{P}(S) = (2^S \setminus \{\emptyset\}, \cup, \cdot)$ and $\mathcal{P}_0(S) = (2^S, \cup, \cdot)$ with the multiplication $A \cdot B = \{ab \mid a \in A, b \in B\}$. The latter definition of power semirings only yields P-complete circuit evaluation problems (resp., NC^1-complete expression evaluation problems):

Lemma 4.7. *Let S be a finite semigroup. Then* $\mathsf{CEP}(\mathcal{P}_0(S))$ *is* P*-complete and* $\mathsf{EEP}(\mathcal{P}_0(S))$ *is* NC^1*-complete.*

Proof. For any idempotent $e \in S$, the subsets $\emptyset$ and $\{e\}$ form a copy of $\mathbb{B}_2$. Hence, the result follows from Theorem 4.3. □

Now we will classify the complexity of $\mathsf{CEP}(\mathcal{P}(S))$ and $\mathsf{EEP}(\mathcal{P}(S))$ for arbitrary semigroups S. It is known that in every finite local group S of size n the minimal semigroup ideal is $S^n = SE(S)S$, see [4, Proposition 2.3].

Proposition 4.8. *Let S be a finite semigroup.*

(1) *If S is a local group and solvable, then $\mathcal{P}(S)$ is $\{0,1\}$-free and $\mathcal{P}(S)_\bullet$ is solvable.*
(2) *If S is not a local group, then $\mathbb{B}_2$ is a subsemiring of $\mathcal{P}(S)$.*
(3) *If S is not solvable, then $\mathcal{P}(S)_\bullet$ is not solvable.*

Proof. For (1) let S be a finite solvable local group. By [6, Corollary 2.7] the multiplicative semigroup $\mathcal{P}(S)_\bullet$ is solvable as well. It remains to show that the semiring $\mathcal{P}(S)$ is $\{0,1\}$-free: Towards a contradiction assume that $\mathcal{P}(S)$ is not $\{0,1\}$-free. By Lemma 3.1, there exist non-empty sets $A, B \subseteq S$ such that $A \neq B$, $A \cup B = B$ (hence $A \subsetneq B$), $AB = BA = A^2 = A$ and $B^2 = B$. Hence, B is a subsemigroup of S, which is also a local group, and A is a semigroup ideal in B. Since the minimal semigroup ideal of B is B^n for $n = |B|$ and $B^n = B$, we obtain $A = B$, which is a contradiction.

For (2) assume that S is not a local group. Hence, there exists a local monoid eSe which is not a group and hence contains an idempotent $f \neq e$. The subsemiring $\{\{f\}, \{e, f\}\}$ is isomorphic to $\mathbb{B}_2$.

Finally, we show (3). Notice that the subsemigroup $\{\{s\} \mid s \in S\}$ of singleton sets in $\mathcal{P}(S)_\bullet$ is isomorphic to S. Hence, if S is not solvable, then $\mathcal{P}(S)$ is also not solvable. □

Together with Corollary 1.2 and Corollary 1.4 we obtain:

Theorem 4.9. *Let S be a finite semigroup. If S is a local group and solvable, then* $\mathsf{CEP}(\mathcal{P}(S))$ *belongs to* DET *and* $\mathsf{EEP}(\mathcal{P}(S))$ *belongs to* TC^0*. Otherwise* $\mathsf{CEP}(\mathcal{P}(S))$ *is* P*-complete and* $\mathsf{EEP}(\mathcal{P}(S))$ *is* NC^1*-complete.*

5 CIRCUITS AND EXPRESSIONS OVER $\{0,1\}$-FREE SEMIRINGS

Throughout this section we fix a finite semiring R of size $n = |R|$. Let $E = E(R)$ be the set of multiplicative idempotent elements. The proof of Theorem 1.1 will proceed in three reduction steps.

(1) Reduce $\mathsf{CEP}(R)$ to $\mathsf{CEP}(R_+)$ and $\mathsf{CEP}(\langle R^n \rangle)$.
(2) Reduce $\mathsf{CEP}(\langle R^n \rangle)$ to the restriction of $\mathsf{CEP}(R)$ to *type admitting* circuits.
(3) If R is $\{0,1\}$-free, reduce the restriction of $\mathsf{CEP}(R)$ to type admitting circuits to $\mathsf{CEP}(R_\bullet)$.

The precise reduction types are made specific in the following lemmas. Theorem 1.3 will be proved simultaneously by a similar reduction chain.

5.1 Step 1: Reduction to R_+ and $\langle R^n \rangle$

Recall that $\langle R^n \rangle$ is the subsemiring of R generated by $R^n \subseteq R$. Since $R^n \cdot R^n = R^n$, $\langle R^n \rangle$ is the set of all finite sums of elements from R^n. We will need the following lemma.

Lemma 5.1. *Let R be a finite $\{0,1\}$-free semiring. If R_+ and $R_\bullet$ are local groups, then $|\langle R^n \rangle| = 1$.*

Proof. Let E be the set of multiplicative idempotents and A be the set of additive idempotents in R.

First we show that $|A| = 1$. Clearly, $(A, +)$ is a commutative semigroup in which every element is idempotent (also called *semi-lattice*). For each $a \in A$ the set $a + A$ forms a submonoid of A

with identity a and by assumption $a + A$ is in fact a group. Since any group contains exactly one idempotent element, we must have $a + A = \{a\}$. For any $a, b \in A$ we have $a + b \in (a + A) \cap (b + A)$, which implies $a = a + b = b$ and proves the claim.

Next we show that $E \subseteq A$. Let $e \in E$ be an arbitrary multiplicative idempotent. The set eRe forms a subsemiring of R and a submonoid of $(R, \cdot)$. By assumption, $(eRe, \cdot)$ is a group with identity e. Now let $k \geq 1$ be any number such that $k \cdot e$ is additively idempotent. Then $k \cdot e$ is also a multiplicative idempotent because $(k \cdot e)^2 = k^2 \cdot e = k \cdot e$. Since e is the only idempotent in the multiplicative group eRe and $k \cdot e \in eRe$, we have $k \cdot e = e$ and hence $e \in A$. This implies $E = A = \{e\}$.

Furthermore, for all $s \in R$ we have $ese = (e + e)se = ese + ese$, which means that ese is an additive idempotent and hence $ese = e$.

Finally, we show that $es = se = e$ for all $s \in R$. From $(es)(es) = (ese)s = es$ it follows that es is a multiplicatively idempotent element, which must be e. Similarly one shows that $se = e$. This shows that $R^n = RER = \{e\}$ and therefore $\langle R^n \rangle = \{e\}$ because e is additively and multiplicatively idempotent. □

As a first step, we reduce CEP(R) (resp., TEP(R)) to the two problems CEP(R_+) and CEP($\langle R^n \rangle$) (resp., TEP(R_+) and TEP($\langle R^n \rangle$)). This lemma is useful because $R^n = RER$ and hence every element $r \in R^n$ can be factorized as $r = set$ where $e \in E$. These idempotent elements will later be used to transform the circuit into a type admitting circuit.

Lemma 5.2. *The following holds:*

- CEP(R) *is* AC^0*-Turing-reducible to* CEP(R_+) *and* CEP($\langle R^n \rangle$).
- TEP(R) *is* TC^0*-Turing-reducible to* TEP($\langle R^n \rangle$).

In particular, if R_+ *and* $R_\bullet$ *are local groups, then:*

- CEP(R) *is* AC^0*-Turing-reducible to* CEP(R_+).
- TEP(R) *belongs to* TC^0.

Proof. Let us first prove the statements for circuits. For a circuit C in normal form with the gate set V we define a circuit C_{long} (which is not in normal form) with the gate set $V^{\leq 2n}$ (all sequences of at most $2n$ gates). A gate in C_{long} is a k-tuple $(v_1, \ldots, v_k) \in V^k$ where $1 \leq k \leq 2n$, which evaluates to

$$[(v_1, \ldots, v_k)]_{C_{\text{long}}} = \prod_{i=1}^{k} [v_i]_C. \tag{1}$$

The circuit computes the product in (1) from products of lower gates v_j' of length at most $2n$. The edges of C_{long} are defined as follows: Let $(v_1, \ldots, v_k) \in V^k$ be a gate in C_{long}. If all gates $v_1, \ldots, v_k$ are input gates of C, then

$$(v_1, \ldots, v_k) \to \prod_{i=1}^{k} [v_i]_C.$$

Otherwise let $1 \leq i \leq k$ be such that v_i is not an input gate and the depth of the subcircuit rooted in v_i is maximal.

- If v_i is an addition gate in C with $v_i \to u_i + w_i$ then
$$(v_1, \ldots, v_k) \to (v_1, \ldots, v_{i-1}, u_i, v_{i+1}, \ldots, v_k) + (v_1, \ldots, v_{i-1}, w_i, v_{i+1}, \ldots, v_k).$$
- If v_i is a multiplication gate in C with $v_i \to u_i \cdot w_i$ and $k < 2n$ then
$$(v_1, \ldots, v_k) \to (v_1, \ldots, v_{i-1}, u_i, w_i, v_{i+1}, \ldots, v_k). \tag{2}$$

- If v_i is a multiplication gate in C and $k = 2n$ then
$$(v_1, \ldots, v_{2n}) \to (v_1, \ldots, v_n) \cdot (v_{n+1}, \ldots, v_{2n}).$$

This circuit C_{long} can be constructed in AC^0. One can easily verify that (1) indeed holds. Notice that the set $W := \bigcup_{n \le k \le 2n} V^k$ forms a downwards closed set and therefore defines a subcircuit of C_{long}. All gates in W evaluate to an element of R^n. Since we always expand a gate v_i with maximal height, the depth of C_{long} only increases by a factor of $2n$ in comparison to C.

Let us first assume that R_+ and $R_\bullet$ are local groups. By Lemma 5.1 we have $\langle R^n \rangle = \{a\}$ for a semiring element a. We can therefore turn every gate $v \in W$ into an input gate with $v \to a$. The resulting circuit only contains addition gates, copy gates (see (2)) and input gates. Since every chain of copy gates has length at most n (which is a constant), we can eliminate the copy gates in AC^0. The resulting circuit can be evaluated using the oracle for $\mathsf{CEP}(R_+)$.

Now assume that R_+ or $R_\bullet$ is not a local group. Hence, $\mathsf{CEP}(R_+)$ or $\mathsf{CEP}(R_\bullet)$ is NL-hard by Theorem 4.2 and normal form can be established using oracle access to $\mathsf{CEP}(R_+)$ or $\mathsf{CEP}(R_\bullet)$. Using the $\mathsf{CEP}(\langle R^n \rangle)$-algorithm one evaluates the subcircuit induced by W. The remaining circuit contains no multiplication gates and hence can be evaluated using the $\mathsf{CEP}(R_+)$-algorithm.

For the tree evaluation problem $\mathsf{TEP}(R)$, let $\mathcal{T}$ be the given tree in ancestor representation. By Proposition 4.6 we can assume that the depth of $\mathcal{T}$ is $O(\log |\mathcal{T}|)$. Let C_{long} be the circuit defined above for $\mathcal{T}$, which is indeed computable by a guarded transduction (see the paragraph before Lemma 2.5). After computing the EC-representation of $\mathcal{T}$ using Lemma 2.2, we can apply Lemma 2.5 to obtain the EC-representation of C_{long}, which has depth $O(\log |\mathcal{T}|)$ as well. Again in TC^0 (using Lemma 2.2) we can unfold C_{long} to the tree $\mathcal{T}_{\text{long}}$ in ancestor representation. Normal form can be established in TC^0 by Lemma 2.4. The rest of the proof follows the above argument for the circuit case. Note that $\mathsf{TEP}(R, +)$ belongs to TC^0. □

Hence, for the case that both R_+ and $R_\bullet$ are local groups, Theorem 1.1 is already proven. In the following two Subsections 5.2 and 5.3 we can therefore assume that R_+ or $R_\bullet$ is not a local group. Hence, $\mathsf{CEP}(R_+)$ or $\mathsf{CEP}(R_\bullet)$ is NL-hard by Theorem 4.2. We can therefore solve graph reachability problems using oracle access to $\mathsf{CEP}(R_+)$ or $\mathsf{CEP}(R_\bullet)$.

5.2 Step 2: Reduction to type admitting circuits

For the next step, it is more convenient to evaluate a circuit C over the free semiring $\mathbb{N}[R]$ generated by the set R.[3] Recall that this semiring consists of all mappings $\alpha \colon R^+ \to \mathbb{N}$ with finite and non-empty support, where R^+ consists of all finite non-empty words over the alphabet R.

So, there are two ways to evaluate C: We can evaluate C over R (and this is our main interest) and we can evaluate C over $\mathbb{N}[R]$. In order to distinguish these two ways of evaluation, we write $[\![v]\!]_C \in \mathbb{N}[R]$ for the value of gate $v \in V$ in C, when C is evaluated in $\mathbb{N}[R]$. Let h be the canonical semiring homomorphism from $\mathbb{N}[R]$ to R that evaluates a non-commutative polynomial in the semiring R. Thus, we have $[v]_C = h([\![v]\!]_C)$ for every gate v and $[C] = h([\![C]\!])$.

Definition 5.3. Let $E = E(R)$ be the set of multiplicative idempotents. A *type function* for a normal form circuit C is a mapping $\mathsf{type} \colon V \to E \times E$ such that for each gate $v \in V(C)$ the following conditions hold:

- If $\mathsf{type}(v) = (e, f)$ then $[v]_C \in eRf$.
- If $v \to u + w$ then $\mathsf{type}(v) = \mathsf{type}(u) = \mathsf{type}(w)$.
- If $v \to u \cdot w$, $\mathsf{type}(u) = (e, e')$ and $\mathsf{type}(w) = (f', f)$ then $\mathsf{type}(v) = (e, f)$.

[3] Of course, it is not possible to evaluate a circuit over the free semiring $\mathbb{N}[R]$ in polynomial time, since this might produce a doubly exponential number of monomials of exponential length. Circuit evaluation over $\mathbb{N}[R]$ is only used as a tool in our proofs.

A circuit is called *type admitting* if it admits a type function.

We write $\mathsf{CEP}_t(R)$ ($\mathsf{TEP}_t(R)$) for the restriction of $\mathsf{CEP}(R)$ ($\mathsf{TEP}(R)$) to type admitting circuits (trees). A type function for a circuit guarantees that a gate v with type (e, f) evaluates to an element from eRf without concretely knowing the value of v. This additional structure will be helpful for the final evaluation step in Section 5.3.

Lemma 5.4. *The following hold:*

- $\mathsf{CEP}(\langle R^n \rangle)$ *is* AC^0*-Turing-reducible to* $\mathsf{CEP}_t(R)$ *and* GAP.
- $\mathsf{TEP}(\langle R^n \rangle)$ *is* TC^0*-Turing-reducible to* $\mathsf{TEP}_t(R)$.

Proof. Let us interpret the input circuit C over the free semiring $\mathbb{N}[R]$. Since $R^n = RER$, for each input gate v we can write $[v]_C$ as

$$[v]_C = \sum_{i=1}^{k} s_i e_i t_i = \sum_{i=1}^{k} (s_i e_i) e_i (e_i t_i)$$

for $s_i, t_i \in R$, $e_i \in E$. By factoring out common prefixes and suffixes we can write $[v]_C$ as

$$[v]_C = \sum_{i=1}^{m} (s_i e_i) a_i (e_i t_i)$$

such that $s_i, t_i, a_i \in R$, $e_i \in E$, and the triples $(s_1, e_1, t_1), \dots, (s_m, e_m, t_m)$ are pairwise distinct. Here m is bounded by a fixed constant since the size of the underlying semiring is a constant. We redefine $v \to \sum_{i=1}^{m} (s_i e_i) a_i (e_i t_i)$ (a sum of m monomials of length 5). Thus for all $v \in V$ we have $\mathsf{supp}(\llbracket v \rrbracket_C) \subseteq (RERER)^+ \subseteq RER^+ER$. Let us emphasize again that here and in the following sets like RE or ER are sets of words of length two over the alphabet R.

For a gate $v \in V$ we define the sets $\mathsf{prefix}(\alpha) \subseteq RE$, $\mathsf{suffix}(\alpha) \subseteq ER$, and $\mathsf{ps}(\alpha) \subseteq RE \times ER$ ("ps" for "prefix-suffix") as follows:

- $\mathsf{prefix}(v) = \{se \mid s \in R, e \in E, \mathsf{supp}(\llbracket v \rrbracket_C) \cap seR^* \neq \emptyset\}$,
- $\mathsf{suffix}(v) = \{ft \mid f \in E, t \in R, \mathsf{supp}(\llbracket v \rrbracket_C) \cap R^* ft \neq \emptyset\}$,
- $\mathsf{ps}(v) = \{(se, ft) \mid s, t \in R, e, f \in E, \mathsf{supp}(\llbracket v \rrbracket_C) \cap seR^* ft \neq \emptyset\}$.

We first compute these sets using the GAP-oracle, which is possible due to the following easy observations:

- For an input gate v with $v \to \sum_{i=1}^{m} (s_i e_i) a_i (e_i t_i)$ we have:

$$\begin{aligned} \mathsf{prefix}(v) &= \{s_i e_i \mid 1 \le i \le k\}, \\ \mathsf{suffix}(v) &= \{e_i t_i \mid 1 \le i \le k\}, \\ \mathsf{ps}(v) &= \{(s_i e_i, e_i t_i) \mid 1 \le i \le m\}. \end{aligned}$$

- For an addition gate v with $v \to v_1 + v_2$ we have:

$$\begin{aligned} \mathsf{prefix}(v) &= \mathsf{prefix}(v_1) \cup \mathsf{prefix}(v_2), \\ \mathsf{suffix}(v) &= \mathsf{suffix}(v_1) \cup \mathsf{suffix}(v_2), \\ \mathsf{ps}(v) &= \mathsf{ps}(v_1) \cup \mathsf{ps}(v_2). \end{aligned}$$

- For a multiplication gate v with $v \to v_1 \cdot v_2$ we have:

$$\begin{aligned} \mathsf{prefix}(v) &= \mathsf{prefix}(v_1), \\ \mathsf{suffix}(v) &= \mathsf{suffix}(v_2), \\ \mathsf{ps}(v) &= \mathsf{prefix}(v_1) \times \mathsf{suffix}(v_2). \end{aligned}$$

Using these identities, one can define the sets $\mathsf{prefix}(v)$, $\mathsf{suffix}(v)$ and $\mathsf{ps}(v)$ using graph reachability: se belongs to $\mathsf{prefix}(v)$ if there exists an input gate u such that $se \in \mathsf{prefix}(v)$ and there exists a path from u to v in the circuit such that for every edge (v', u') along this path either v' is an addition gate or v' is a multiplication gate and the u' is the left successor of v'. The set $\mathsf{suffix}(v)$ can be obtained analogously. Moreover, we have $(se, ft) \in \mathsf{ps}(v)$ if there exists a (possibly empty) path from v to a gate u such that every gate $v' \neq u$ along the path is an addition gate and one of the following two cases holds:

- u is an input gate and $(se, ft) \in \mathsf{ps}(u)$.
- u is a multiplication gate, $se \in \mathsf{prefix}(u)$, and $ft \in \mathsf{suffix}(u)$.

Now we will construct in logspace a circuit $\mathcal{D}$ containing for each $v \in V(C)$ with $(se, ft) \in \mathsf{ps}(v)$ a gate $v_{s,e,f,t}$ such that

$$[\![v]\!]_C = \sum_{(se,ft)\in \mathsf{ps}(v)} s \cdot [\![v_{s,e,f,t}]\!]_{\mathcal{D}} \cdot t \tag{3}$$

and every monomial in $[\![v_{s,e,f,t}]\!]_{\mathcal{D}}$ is from eR^*f.

Case 1. If v is an input gate with $v \to \sum_{i=1}^{m} (s_i e_i) a_i (e_i t_i)$ as above, then we set

$$v_{s_i,e_i,e_i,t_i} \to e_i a_i e_i.$$

Case 2. If $v \to u \cdot w$, we set

$$v_{s,e,f,t} \to \sum_{(se,f't')\in \mathsf{ps}(u)} \; \sum_{(s'e',ft)\in \mathsf{ps}(w)} u_{s,e,f',t'} \cdot (f't's'e') \cdot w_{s',e',f,t}.$$

Note that the right-hand side of this definition is a sum of a constant number of products since we deal with a fixed semiring.

Case 3. If $v \to u + w$, we set

$$v_{s,e,f,t} \to \begin{cases} u_{s,e,f,t} + w_{s,e,f,t} & \text{if } (s,e,f,t) \in \mathsf{ps}(u) \cap \mathsf{ps}(w), \\ u_{s,e,f,t} & \text{if } (s,e,f,t) \in \mathsf{ps}(u) \setminus \mathsf{ps}(w), \\ w_{s,e,f,t} & \text{if } (s,e,f,t) \in \mathsf{ps}(w) \setminus \mathsf{ps}(u). \end{cases}$$

It is easy to verify that (3) holds and that every monomial in $[\![v_{s,e,f,t}]\!]_{\mathcal{D}}$ belongs to eR^*f. Hence, one can define a type function on $\mathcal{D}$ by $\mathsf{type}(v_{s,e,f,t}) = (e, f)$. Moreover, using Lemma 2.3 we transform $\mathcal{D}$ in logspace into normal form. Thereby we extend the type-mapping to the new gates that are introduced. For example, all partial sums in Case 2 get type (e, f) and the product $(f't's'e')$ gets type (f', e'). Once the circuit $\mathcal{D}$ is evaluated in the semiring R (using an oracle for the restriction of $\mathsf{CEP}(R)$ to type admitting circuits) the circuit C can be evaluated in AC^0 using (3). This last step only involves a constant number of semiring operations.

The statement for trees is proved using the same arguments as in the proof of Lemma 5.2, since (i) one can again assume a logarithmic depth input tree over $\langle R^n \rangle$ using Proposition 4.6, and (ii) the transformation $C \mapsto \mathcal{D}$ above is a guarded transduction that preserves the depth up to a constant factor. Also note that the three sets $\mathsf{prefix}(v)$, $\mathsf{suffix}(v)$ and $\mathsf{ps}(v)$ can be computed in AC^0 (without oracle access to GAP) for every node of the input tree since we assume that the ancestor relation is available. □

5.3 Step 3: A parallel evaluation algorithm for type admitting circuits and trees

In this section we present an evaluation algorithm for type admitting circuits and trees. This algorithm terminates after at most $|R|$ rounds, if R has a so-called rank-function, which we define first. As before, let $E = E(R)$.

Definition 5.5. We call a function $\mathsf{rank}\colon R \to \mathbb{N} \setminus \{0\}$ a *rank-function* for R if it satisfies the following conditions for all $a, b \in R$:

(1) $\mathsf{rank}(a) \leq \mathsf{rank}(a + b)$
(2) $\mathsf{rank}(a), \mathsf{rank}(b) \leq \mathsf{rank}(a \cdot b)$
(3) If $a, b \in eRf$ for some $e, f \in E$ and $\mathsf{rank}(a) = \mathsf{rank}(a + b)$, then $a = a + b$.

Note that if $R_\bullet$ is a monoid, then one can choose $e = 1 = f$ in the third condition in Definition 5.5, which is therefore equivalent to: If $\mathsf{rank}(a) = \mathsf{rank}(a + b)$ for $a, b \in R$, then $a = a + b$.

Example 5.6. Let G be a finite group and consider the power semiring $\mathcal{P}(G)$ defined in Section 4.3. One can verify that the function $A \mapsto |A|$, where $\emptyset \neq A \subseteq G$, is a rank-function for $\mathcal{P}(G)$. On the other hand, if S is a finite semigroup, which is not a group, then S cannot be cancellative. Assume that $ab = ac$ for $a, b, c \in S$ with $b \neq c$. Then $\{a\} \cdot \{b, c\} = \{ab\}$. This shows that the function $A \mapsto |A|$ is not a rank-function for $\mathcal{P}(S)$.

Lemma 5.7. *Let R be either $\mathbb{B}_2$ or $\mathbb{Z}_q$ for some $q \geq 2$. Then R does not have a rank-function.*

Proof. Assume that rank is a rank-function for R and let $0 \neq 1$ be the additive and multiplicative units in R. The first two properties of a rank-function imply that $\mathsf{rank}(0) \leq \mathsf{rank}(0+1) = \mathsf{rank}(1) \leq \mathsf{rank}(1 \cdot 0) = \mathsf{rank}(0)$, and hence $\mathsf{rank}(0) = \mathsf{rank}(0 + 1)$. Since $e := 1$ is multiplicatively idempotent and $0, 1 \in eRe = R$, we know $0 = 0 + 1 = 1$ by the third property of a rank-function. □

Lemma 5.8. *If R is a finite semiring which is not $\{0, 1\}$-free, then it does not have a rank-function.*

Proof. By Lemma 3.1 R contains a subsemiring T which is either $\mathbb{B}_2$ or $\mathbb{Z}_q$ for some $q \geq 2$. If rank would be a rank-function for R, then its restriction to T would be a rank-function for T, contradicting Lemma 5.7. □

Theorem 5.9. *Assume that R has a rank-function.*

(1) *$\mathsf{CEP}_t(R)$ is AC^0-Turing-reducible to $\mathsf{CEP}(R_\bullet)$ and GAP.*
(2) *$\mathsf{TEP}_t(R)$ is TC^0-Turing-reducible to $\mathsf{WP}(R_\bullet)$.*

Proof. Let C be a type admitting circuit. We present an algorithm which partially evaluates the circuit in a constant number of phases, where each phase can be carried out in AC^0 with oracle access to $\mathsf{CEP}(R_\bullet)$ and GAP. The following invariant is preserved:

Invariant. *After phase k all gates A with $\mathsf{rank}([A]_C) \leq k$ are evaluated, i.e., are input gates from phase $k + 1$ onwards.*

In the beginning, i.e., for $k = 0$, the invariant clearly holds (since 0 is not in the range of the rank-function). After $\max\{\mathsf{rank}(a) \mid a \in R\}$ (which is a constant) many phases, all gates are evaluated. We present phase k of the algorithm, assuming that the invariant holds after phase $k - 1$. Thus, all gates v with $\mathsf{rank}([v]_C) < k$ of the current circuit C are input gates. The goal of phase k is to evaluate all gates v with $\mathsf{rank}([v]_C) = k$. For this, we define the circuit $C_\bullet$ over $R_\bullet$ with $V(C_\bullet) = V(C)$: If $v \in V(C)$ is a multiplication or an input gate, then v has the same incoming wires in $C_\bullet$ as in C. If $v \in V(C)$ is an addition gate with $v \to u + w$ in C, then in the circuit $C_\bullet$ we set

$$v \to \begin{cases} a + b & \text{if } u \to a \in R \text{ and } w \to b \in R, \\ u & \text{if } u \text{ is an inner gate,} \\ w & \text{if } u \text{ is not inner but } w \text{ is an inner gate,} \end{cases} \tag{4}$$

i.e., v is directly evaluated if both of its input gates are already evaluated and otherwise it copies the value of one of its inner input gates (our priority choice for u is arbitrary). Note that the value of such an inner gate has rank at least k because it is not evaluated yet. The circuit $\mathcal{C}_\bullet$ can be brought into normal form by Lemma 2.3 and then evaluated using the oracle for CEP($R_\bullet$). A gate $v \in V$ is called *locally correct* if (i) v is an input gate or a multiplication gate of $\mathcal{C}$, or (ii) v is an addition gate of $\mathcal{C}$ with $v \to u + w$ and $[v]_{\mathcal{C}_\bullet} = [u]_{\mathcal{C}_\bullet} + [w]_{\mathcal{C}_\bullet}$. We compute the set

$$W := \{v \in V \mid \text{all gates } w \leq_{\mathcal{C}} v \text{ are locally correct}\} \tag{5}$$

using the GAP-algorithm. A simple induction shows that for all $v \in W$ we have $[v]_{\mathcal{C}} = [v]_{\mathcal{C}_\bullet}$. Hence we can set $v \to [v]_{\mathcal{C}_\bullet}$ in $\mathcal{C}$ for all $v \in W$. This concludes phase k of the algorithm.

To prove that the invariant still holds after phase k, we show that for each gate $v \in V$ with $\mathsf{rank}([v]_{\mathcal{C}}) \leq k$ we have $v \in W$. This is shown by induction over the depth of v in $\mathcal{C}$. Assume that $\mathsf{rank}([v]_{\mathcal{C}}) \leq k$. By the first two conditions from Definition 5.5, all gates $w <_{\mathcal{C}} v$ satisfy $\mathsf{rank}([w]_{\mathcal{C}}) \leq k$. Thus, the induction hypothesis yields $w \in W$ and hence $[w]_{\mathcal{C}} = [w]_{\mathcal{C}_\bullet}$ for all gates $w <_{\mathcal{C}} v$.

It remains to show that v is locally correct, which is clear if v is an input gate, a multiplication gate or an addition gate whose input gates are input gates of $\mathcal{C}$. So assume that $v \to u+w$ where w.l.o.g. u is an inner gate, which implies $[v]_{\mathcal{C}_\bullet} = [u]_{\mathcal{C}_\bullet}$ by (4). Since u is an inner gate, which is not evaluated after phase $k-1$, it holds that $\mathsf{rank}([u]_{\mathcal{C}}) \geq k$ and therefore $k \geq \mathsf{rank}([v]_{\mathcal{C}}) \geq \mathsf{rank}([u]_{\mathcal{C}}) \geq k$, i.e., $\mathsf{rank}([v]_{\mathcal{C}}) = \mathsf{rank}([u]_{\mathcal{C}}) = k$. Since $\mathcal{C}$ is type admitting, there exist idempotents $e, f \in E$ with $[u]_{\mathcal{C}}, [w]_{\mathcal{C}} \in eRf$. The third condition from Definition 5.5 implies that $[v]_{\mathcal{C}} = [u]_{\mathcal{C}} + [w]_{\mathcal{C}} = [u]_{\mathcal{C}}$. We get

$$[v]_{\mathcal{C}_\bullet} = [u]_{\mathcal{C}_\bullet} = [u]_{\mathcal{C}} = [v]_{\mathcal{C}} = [u]_{\mathcal{C}} + [w]_{\mathcal{C}} = [u]_{\mathcal{C}_\bullet} + [w]_{\mathcal{C}_\bullet}.$$

Therefore v is locally correct.

The second statement concerning trees is clear: If $\mathcal{T}$ is a tree then $\mathcal{T}_\bullet$ is a tree, which is TC^0-computable. To see that the ancestor relation of $\mathcal{T}_\bullet$ is TC^0-computable, note that v is an ancestor of u in $\mathcal{T}_\bullet$ if v is an ancestor of u in $\mathcal{T}$ and for every multiplication gate $w \neq v$ along the unique $\mathcal{T}$-path from u to v one of the following two cases holds: (i) both $\mathcal{T}$-children of w are input gates, (ii) the unique child w' of w along the path to v is an inner node of $\mathcal{T}$ and if the other child of w is also an inner node, then w' is the left child of w. Normal form of $\mathcal{T}_\bullet$ can be established in TC^0 by Lemma 2.3. Once $\mathcal{T}_\bullet$ is evaluated using the oracle for WP($R_\bullet$), the set W from (5) is TC^0-computable using the ancestor relation of $\mathcal{T}$. □

Example 5.10 (Example 5.6 continued). Figure 2 shows a circuit $\mathcal{C}$ over the power semiring $\mathcal{P}(G)$ of the group $G = (\mathbb{Z}_5, +)$. Recall from Example 5.6 that the function $A \mapsto |A|$ is a rank function for $\mathcal{P}(G)$. We illustrate one phase of the algorithm. All gates A with $\mathsf{rank}([A]) < 3$ are evaluated in the circuit $\mathcal{C}$ shown in (a). The goal is to evaluate all gates A with $\mathsf{rank}([A]) = 3$. The circuit $\mathcal{C}_\bullet$ (shown in (b)) from the proof of Theorem 5.9 is computed and evaluated using the oracle for CEP($\mathbb{Z}_5, +$). The dotted wires do not belong to the circuit $\mathcal{C}_\bullet$. All locally correct gates are shaded. The shaded gates form a downwards closed set, which is the set W from (5). These gates can be evaluated such that in the resulting circuit (shown in (c)) all gates which evaluate to elements of rank 3 are evaluated.

It remains to show that every finite $\{0, 1\}$-free semiring has a rank-function.

Lemma 5.11. *Let R be $\{0, 1\}$-free. If $e, f \in E$ and $f = ef = fe = f + f$, then $e + f = f$.*

Proof. Setting $0 := f$ and $1 := e + f$ we have $0 + 0 = 0$, $0 + 1 = 1$, $0 \cdot 1 = 1 \cdot 0 = 0 \cdot 0 = 0$, and $1 \cdot 1 = 1$. Since R is $\{0, 1\}$-free, Lemma 3.1 (point 4) implies that $0 = 1$, i.e. $e + f = f$. □

Lemma 5.12. *Let R be a finite semiring. Then R is $\{0, 1\}$-free if and only if R has a rank-function.*

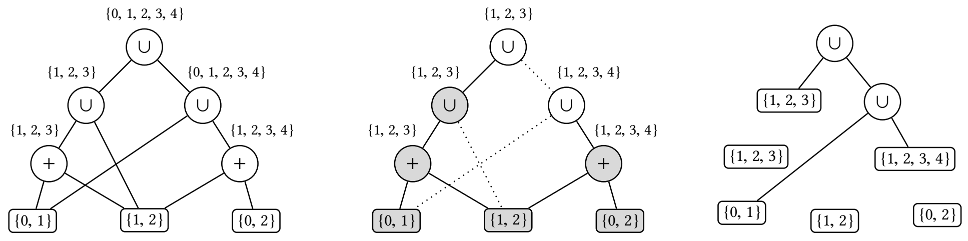

Fig. 2. The parallel evaluation algorithm over the power semiring $\mathcal{P}(\mathbb{Z}_5)$.

Proof. The "if"-direction is Lemma 5.8. For the "only if"-direction assume that R is $\{0,1\}$-free. For $a, b \in R$ we define $a \preceq b$ if b can be obtained from a by iterated additions and left- and right-multiplications of elements from R. This is equivalent to the following condition:

$$\exists \ell, r, c \in R : b = \ell a r + c \text{ (where each of the elements } \ell, r, c \text{ can be also missing)}$$

Since $\preceq$ is a preorder on R, there is a function $\mathsf{rank} : R \to \mathbb{N} \setminus \{0\}$ such that for all $a, b \in R$ we have

- $\mathsf{rank}(a) = \mathsf{rank}(b)$ iff $a \preceq b$ and $b \preceq a$,
- $\mathsf{rank}(a) \leq \mathsf{rank}(b)$ if $a \preceq b$.

We claim that rank satisfies the conditions of Definition 5.5. The first two conditions are clear, since $a \preceq a + b$ and $a, b \preceq ab$. For the third condition, let $e, f \in E$, $a, b \in eRf$ such that $\mathsf{rank}(a + b) = \mathsf{rank}(a)$, which is equivalent to $a + b \preceq a$. Assume that $a = \ell(a + b)r + c = \ell a r + \ell b r + c$ for some $\ell, r, c \in R$ (the case without c can be handled in the same way). Since $a, b \in eRf$ we know $a = eaf$ and $b = ebf$, and therefore $a = \ell e(a + b) f r + c$. Hence we can assume that ℓ and r are not missing. By multiplying with e from the left and f from the right we get $a = (e\ell e)(a + b)(frf) + (ecf)$, so we can assume that $\ell = e\ell e$ and $r = frf$. After m repeated applications of $a = \ell a r + \ell b r + c$ we obtain

$$a = \ell^m a r^m + \sum_{i=1}^{m} \ell^i b r^i + \sum_{i=0}^{m-1} \ell^i c r^i. \tag{6}$$

Let $n \geq 1$ such that nx is additively idempotent and x^n is multiplicatively idempotent for all $x \in R$. Hence nx^n is both additively and multiplicatively idempotent for all $x \in R$. If we choose $m = n^2$, the right hand side of (6) contains the partial sum $\sum_{i=1}^{n} \ell^{in} b r^{in}$. Furthermore, $e(n\ell^n) = (n\ell^n)e = n\ell^n$ and $f(nr^n) = (nr^n)f = nr^n$. Therefore, Lemma 5.11 implies that $n\ell^n = n\ell^n + e$ and $nr^n = nr^n + f$, and hence:

$$\begin{aligned}\sum_{i=1}^{n} \ell^{in} b r^{in} &= n(\ell^n b r^n) = n^2(\ell^n b r^n) = (n\ell^n)b(nr^n) = (n\ell^n + e)b(nr^n) \\ &= (n\ell^n)b(nr^n) + eb(nr^n) = (n\ell^n)b(nr^n) + eb(nr^n + f) \\ &= (n\ell^n)b(nr^n) + eb(nr^n) + ebf = \left(\sum_{i=1}^{n} \ell^{in} b r^{in}\right) + b.\end{aligned}$$

Thus, we can replace in (6) the partial sum $\sum_{i=1}^{n} \ell^{in} b r^{in}$ by $\sum_{i=1}^{n} \ell^{in} b r^{in} + b$, which proves that $a = a + b$. □

We remark that if a finite semiring R is not $\{0,1\}$-free, then it cannot possess a rank-function:

Proof of Theorem 1.1 and Theorem 1.3. The case that R_+ and $R_\bullet$ are local groups is clear, see the comment following the proof of Lemma 5.2. For the case R_+ or $R_\bullet$ is not a local group we carry out the reductions from Lemma 5.2 and Lemma 5.4, followed by Theorem 5.9. In each step, oracle access to GAP can be replaced by oracle access to $\mathsf{CEP}(R_+)$ or $\mathsf{CEP}(R_\bullet)$ (one of them is NL-hard). □

6 APPLICATIONS TO FORMAL LANGUAGE THEORY

In this section we present applications of our main results to intersection problems from formal language theory. More precisely, we will apply Theorem 1.1 on circuit evaluation to the problem whether a given context-free language intersects a fixed regular language. Theorem 1.3 on expression evaluation is applied to the problem whether the language produced by a given regular language intersects a fixed regular language.

6.1 Intersection problem for context-free grammars

Recall that a *context-free grammar* (over Σ) is a tuple $\mathcal{G} = (V, \Sigma, S, P)$ consisting of a finite set of variables V, a finite alphabet Σ, a start variable $S \in V$ and a set P of productions $A \to \alpha$ where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$. We write $L_{\mathcal{G}}(A)$ for the language of $A \in V$, i.e., the set of words $w \in \Sigma^*$ which can be derived from A using the productions in P, and write $L(\mathcal{G})$ for $L_{\mathcal{G}}(S)$. If $L_{\mathcal{G}}(A) \neq \emptyset$, we say that A is *productive*. If the start variable S is productive, then $\mathcal{G}$ is called *productive*. Every circuit over the free monoid Σ^* can be seen as a context-free grammar producing exactly one word. Such a circuit is also called a *straight-line program*, briefly SLP. It is a context-free grammar $\mathcal{H} = (V, \Sigma, S, P)$ that contains for every variable $A \in V$ exactly one rule of the form $A \to \alpha$. Moreover, $\mathcal{H}$ is acyclic, i.e., there is no non-empty derivation from a variable A to a word containing A. We denote with $\mathsf{val}_{\mathcal{H}}(A)$ the unique word in the language $L_{\mathcal{H}}(A)$. Moreover, let $\mathsf{val}(\mathcal{H}) = \mathsf{val}_{\mathcal{H}}(S)$.

Given an alphabet Σ and a language $L \subseteq \Sigma^*$, the CFG-*intersection non-emptiness problem for* L, denoted by $\mathsf{CFG\text{-}IP}(L, \Sigma)$, is the following decision problem:

Input A context-free grammar $\mathcal{G}$ over Σ
Question Does $L(\mathcal{G}) \cap L \neq \emptyset$ hold?

For every regular language L, this problem is solvable in polynomial-time, which seems to be folklore but we will reprove this fact in the following. The standard proof constructs a context-free grammar for $L(\mathcal{G}) \cap L$ from the given grammar $\mathcal{G}$ and a finite automaton for L. The constructed grammar then has to be tested for emptiness, which is possible in polynomial time. However, testing emptiness of a given context-free language is P-complete [24]. An easy reduction shows that the problem $\mathsf{CFG\text{-}IP}(L, \Sigma)$ is P-hard for any non-empty language L.

Lemma 6.1. *For every non-empty language $L \subseteq \Sigma^*$ the problem* $\mathsf{CFG\text{-}IP}(L, \Sigma)$ *is* P*-hard.*

Proof. Let $\mathcal{G} = (V, \Sigma, S, P)$ be a context-free grammar. We reduce emptiness of $\mathcal{G}$ to the intersection non-emptiness problem as follows. Let $X \notin V$ be a new variable. We replace all occurrences of terminal symbols in productions of $\mathcal{G}$ by X and then add the rules $X \to \varepsilon$ and $X \to aX$ for all $a \in \Sigma$ (thus, X produces Σ^*). Observe that the new grammar $\mathcal{G}'$ satisfies $L(\mathcal{G}) \neq \emptyset$ if and only if $L(\mathcal{G}') \neq \emptyset$. Further, $L(\mathcal{G}')$ is either $\emptyset$ or Σ^*. Hence, $L(\mathcal{G}) \neq \emptyset$ if and only if $L(\mathcal{G}') \cap L \neq \emptyset$. Clearly, the reduction can be performed in logspace. □

By Lemma 6.1 we have to put some restriction on context-free grammars in order to get NC-algorithms for the intersection non-emptiness problem. It turns out that productivity of all variables is the right assumption. Thus, we require that $L_{\mathcal{G}}(A) \neq \emptyset$ for all $A \in V$. In order to avoid a promise problem (testing productivity of a variable is P-complete) we add to the input grammar $\mathcal{G} = (V, \Sigma, S, P)$ an SLP $\mathcal{H} = (V, \Sigma, S, R)$ which *uniformizes* $\mathcal{G}$ in the sense that R contains for every variable $A \in V$ exactly one rule $(A \to \alpha) \in P$. Hence, the word $\mathsf{val}_{\mathcal{H}}(A) \in L_{\mathcal{G}}(A)$ is a witness for $L_{\mathcal{G}}(A) \neq \emptyset$.

Example 6.2. Here is a context-free grammar, where the underlined productions form a uniformizing SLP:

$$S \to SS,\ S \to aSb,\ \underline{S \to A},\ A \to aA,\ \underline{A \to B},\ B \to bB,\ \underline{B \to b}$$

We study the following decision problem PCFG-IP(L, Σ) in the rest of this section:

Input A productive context-free grammar $\mathcal{G}$ over Σ and a uniformizing SLP $\mathcal{H}$ for $\mathcal{G}$.
Question Does $L(\mathcal{G}) \cap L \neq \emptyset$ hold?

The goal of this section is to classify regular languages L by the complexity of PCFG-IP(L, Σ).

6.1.1 Reduction to circuit evaluation. In the following we prove that PCFG-IP(L, Σ) is equivalent (with respect to AC^0-Turing-reductions) to the circuit evaluation problem for a suitable finite semiring that is derived from L.

We start with a few standard notations from algebraic language theory. A language $L \subseteq \Sigma^*$ is *recognized* by a monoid M if there exists a homomorphism $h\colon \Sigma^* \to M$ such that $h^{-1}(F) = L$ for some $F \subseteq M$. It is known that a language is regular if and only if it is recognized by a finite monoid. The *syntactic congruence* $\equiv_L$ is the equivalence relation on Σ^* that is defined by $u \equiv_L v$ $(u, v \in \Sigma^*)$ if the following holds: $\forall x, y \in \Sigma^* : xuy \in L \Leftrightarrow xvy \in L$. It is indeed a congruence relation on the free monoid Σ^*. The quotient monoid $\Sigma^*/{\equiv_L}$ is the smallest monoid which recognizes L; it is called the *syntactic monoid of* L. From now on we fix a regular language $L \subseteq \Sigma^*$, a surjective homomorphism $h\colon \Sigma^* \to M$ onto the finite syntactic monoid M of L and a set $F \subseteq M$ satisfying $h^{-1}(F) = L$.

Given an algebraic structure $\mathcal{A}$ with domain $A \subseteq 2^M$ (which will be either $\mathcal{P}(M)$ or $\mathcal{P}_0(M)$; see Section 4.3) we define the decision problem CEP$(\mathcal{A}, F)$:

Input A circuit C over $\mathcal{A}$
Question Does $[C] \cap F \neq \emptyset$ hold?

Lemma 6.3. CFG-IP(L, Σ) *is* AC^0*-many-one-reducible to* CEP$(\mathcal{P}_0(M), F)$.

Proof. Let $h\colon \Sigma^* \to M$ be the syntactic homomorphism and let $\mathcal{G} = (V, \Sigma, S, P)$ be a context-free grammar. To decide whether $L(\mathcal{G}) \cap L \neq \emptyset$, we construct a circuit whose gates compute all sets $X_A = h(L_{\mathcal{G}}(A)) \in \mathcal{P}_0(M)$ for $A \in V$. Then we test whether X_S intersects F.

The tuple $(X_A)_{A \in V}$ is the least fixed-point of the following monotone operator μ:

$$\mu\colon \left(2^M\right)^{|V|} \to \left(2^M\right)^{|V|} \tag{7}$$

$$\mu((Y_A)_{A\in V}) = \left(Y_A \cup \bigcup h(\alpha_0) Y_{A_1} h(\alpha_1) \cdots Y_{A_k} h(\alpha_k)\right)_{A \in V} \tag{8}$$

where the union in (8) ranges over all productions $A \to \alpha_0 A_1 \alpha_1 \cdots A_k \alpha_k \in P$ for $A_1, \dots, A_k \in V$ and $\alpha_0, \dots, \alpha_k \in \Sigma^*$. The smallest fixed-point of μ can be computed by the fixed-point iteration

$$Y_A^{(0)} = \emptyset, \quad Y_A^{(n+1)} = \mu((Y_A^{(n)})_{A \in V}) \tag{9}$$

which reaches the least fixed-point after at most $|V| \cdot |M|$ steps. Equation (9) gives rise to an AC^0-computable circuit over the semiring $\mathcal{P}_0(M)$ computing $X_S = h(L(\mathcal{G}))$. □

Corollary 6.4. *For every regular language* $L \subseteq \Sigma^*$ *the problem* CFG-IP(L, Σ) *is in* P.

Lemma 6.5. PCFG-IP(L, Σ) *is equivalent to* CEP$(\mathcal{P}(M), F)$ *with respect to* AC^0*-many-one-reductions.*

Proof. For the reduction from PCFG-IP(L, Σ) to CEP$(\mathcal{P}(M), F)$ we only give the modifications for the proof of Lemma 6.3. We are given a productive context-free grammar $\mathcal{G} = (V, \Sigma, S, P)$ and a uniformizing SLP $\mathcal{H} = (V, \Sigma, S, R)$ for $\mathcal{G}$. We modify the fixed point iteration in (9) as follows: Instead of initializing the sets $Y_A^{(0)}$ with the empty set, we start the fixed point iteration with $Y_A^{(0)} := \{h(\mathrm{val}_{\mathcal{H}}(A))\}$ for $A \in V$. For this we compute from $\mathcal{H}$ a circuit whose gates evaluate to the singleton sets $\{h(\mathrm{val}_{\mathcal{H}}(A))\}$ for $A \in V$. Every production $(A \to \alpha_0 A_1 \alpha_1 \cdots A_k \alpha_k) \in R$ with $A_1, \dots, A_k \in V$ and $\alpha_0, \dots, \alpha_k \in \Sigma^*$ is translated to the definition

$$Y_A^{(0)} \to \{h(\alpha_0)\} \cdot Y_{A_1}^{(0)} \cdot \{h(\alpha_1)\} \cdots Y_{A_k}^{(0)} \cdot \{h(\alpha_k)\}$$

Let us now reduce CEP$(\mathcal{P}(M), F)$ to PCFG-IP(L, Σ). Let $C = (V, E, \lambda, v_0)$ be a circuit over $\mathcal{P}(M)$. We define a grammar $\mathcal{G} = (V, \Sigma, v_0, P)$ as follows:

- If $v \to \{m_1, \dots, m_k\} \in \mathcal{P}(M)$, add the rules $A \to w_i$ to P $(1 \le i \le k)$ where $w_i \in \Sigma^*$ is any word with $h(w_i) = m_i$.
- If $v \to u \cup w$, add the rules $v \to u$ and $v \to w$ to P.
- If $v \to u \cdot w$, add the rules $v \to uw$ to P.

Then every gate $v \in V$ evaluates to $h(L_{\mathcal{G}}(v))$. In particular, we have $h(L(\mathcal{G})) = [C]$. Therefore, $[C] \cap F \neq \emptyset$ if and only if $L(\mathcal{G}) \cap L \neq \emptyset$. □

Now clearly CEP$(\mathcal{P}(M), F)$ is AC^0-Turing-reducible to CEP$(\mathcal{P}(M))$ but not necessarily vice versa (assuming NL $\neq$ P) as the following example shows:

Example 6.6. Consider the language $L = \{a, b\}^* a \{a, b\}^* \subseteq \{a, b\}^*$ of all words which contain the symbol a. Its syntactic monoid is the two-element monoid $M = \{1, e\}$ where e is an idempotent element. We have $L = h^{-1}(\{e\})$ for the homomorphism $h \colon \{a, b\}^* \to M$ defined by $h(a) = e$, $h(b) = 1$. One can decide CEP$(\mathcal{P}(M), \{e\})$ in NL: For a circuit C we have $e \in [C]$ if and only if an input gate with e on the right-hand side is reachable from the output gate. However, since M is not a local group, CEP$(\mathcal{P}(M))$ is P-complete by Theorem 4.9. This can be also seen directly: The sets $\{e\}$ and $\{1, e\}$ form a Boolean semiring. On the other hand, for the purpose of deciding CEP$(\mathcal{P}(M), \{e\})$ one does not have to distinguish the sets $\{e\}$ and $\{1, e\}$. Identifying these two sets in $\mathcal{P}(M)$ yields a $\{0, 1\}$-free semiring whose circuit evaluation problem is in NL.

The example above motivates to define a congruence relation on $\mathcal{P}(M)$ where congruent subsets are either both disjoint from F or both not. Define the equivalence relation $\sim_F$ on $\mathcal{P}(M)$ by

$$A_1 \sim_F A_2 \iff \forall \ell, r \in M \colon \ell A_1 r \cap F \neq \emptyset \iff \ell A_2 r \cap F \neq \emptyset$$

for subsets $A_1, A_2 \in \mathcal{P}(M)$. The following lemma summarizes the basic properties of $\sim_F$.

Lemma 6.7. *The following properties hold.*

(1) *$A_1 \sim_F A_2$ implies $(LA_1R \cap F \neq \emptyset \iff LA_2R \cap F \neq \emptyset)$ for all $L, R \subseteq M$.*
(2) *The relation $\sim_F$ is a semiring congruence on $\mathcal{P}(M)$. In particular, the quotient $\mathcal{P}(M)/{\sim_F}$ is a semiring.*
(3) *Every $\sim_F$-class contains a largest subset with respect to $\subseteq$.*

Proof. Property (1) is clear because $LA_iR \cap F \neq \emptyset$ if and only if $\ell A_i r \cap F \neq \emptyset$ for some $\ell \in L, r \in R$. For (2), assume $A_1 \sim_F A_2$ and $B_1 \sim_F B_2$. Then for all $\ell, r \in M$ we have

$$\begin{aligned} \ell(A_1 \cup B_1)r \cap F \neq \emptyset &\iff (\ell A_1 r \cup \ell B_1 r) \cap F \neq \emptyset \\ &\iff \ell A_1 r \cap F \neq \emptyset \quad \text{or} \quad \ell B_1 r \cap F \neq \emptyset \\ &\iff \ell A_2 r \cap F \neq \emptyset \quad \text{or} \quad \ell B_2 r \cap F \neq \emptyset \\ &\iff (\ell A_2 r \cup \ell B_2 r) \cap F \neq \emptyset \\ &\iff \ell(A_2 \cup B_2)r \cap F \neq \emptyset \end{aligned}$$

and, by (1),

$$\ell A_1(B_1 r) \cap F \neq \emptyset \iff (\ell A_2)B_1 r \cap F \neq \emptyset \iff \ell A_2 B_2 r \cap F \neq \emptyset.$$

For (3) note that by (2), $A_1 \sim_F A_2$ implies $A_1 = A_1 \cup A_1 \sim_F A_1 \cup A_2$. Thus, every $\sim_F$-class is closed under union and therefore has a largest element with respect to $\subseteq$. □

Lemma 6.8. *$\mathsf{CEP}(\mathcal{P}(M), F)$ is equivalent to $\mathsf{CEP}(\mathcal{P}(M)/{\sim_F})$ with respect to AC^0-Turing-reductions.*

Proof. Clearly, every circuit C over $\mathcal{P}(M)$ can be regarded as a circuit C' over $\mathcal{P}(M)/{\sim_F}$ such that $[C']$ is the $\sim_F$-class of $[C]$. Every $\sim_F$-class either contains only subsets of M which are disjoint to F or only subsets with non-empty intersection with F. Thus, $[C']$ determines whether $[C] \cap F \neq \emptyset$.

For the other direction, given a circuit C' over $\mathcal{P}(M)/{\sim_F}$, we define a circuit C over $\mathcal{P}(M)$ by picking arbitrary representative elements (subsets of M) for the input values (which are $\sim_F$-classes) of the circuit C'. Then we test for all $\ell, r \in M$ whether $\ell[C]r \cap F \neq \emptyset$. This information is independent from the choice of representative elements and uniquely determines the $\sim_F$-class $[C']$. □

From Corollary 1.2, Lemma 6.5 and Lemma 6.8 we obtain:

Theorem 6.9. *$\mathsf{PCFG\text{-}IP}(L, \Sigma)$ is equivalent to $\mathsf{CEP}(\mathcal{P}(M)/{\sim_F})$ with respect to AC^0-Turing-reductions. Therefore,*

- *$\mathsf{PCFG\text{-}IP}(L, \Sigma)$ is P-complete if $\mathcal{P}(M)/{\sim_F}$ is not $\{0, 1\}$-free or its multiplicative semigroup is not solvable,*
- *$\mathsf{PCFG\text{-}IP}(L, \Sigma)$ is in DET if $\mathcal{P}(M)/{\sim_F}$ is $\{0, 1\}$-free and its multiplicative semigroup is solvable, and*
- *$\mathsf{PCFG\text{-}IP}(L, \Sigma)$ is in NL if $\mathcal{P}(M)/{\sim_F}$ is $\{0, 1\}$-free and its multiplicative semigroup is aperiodic.*

It would be nice to have a simple characterization of when $\mathcal{P}(M)/{\sim_F}$ is $\{0, 1\}$-free (resp., its multiplicative semigroup is solvable). For $\{0, 1\}$-freeness, we can show:

Proposition 6.10. *$\mathcal{P}(M)/{\sim_F}$ is $\{0, 1\}$-free if and only if*

$$\forall s, t \in M,\ e \in E(M) : st \in F \implies set \in F. \tag{10}$$

Proof. Assume first that $\mathcal{P}(M)/{\sim_F}$ is $\{0, 1\}$-free. Let $s, t \in M$ such that $st \in F$ and let $e \in E(M) \setminus \{1\}$. We have $\{e\} \sim_F \{1, e\}$ since otherwise their two $\sim_F$-classes would form a Boolean subsemiring $\mathbb{B}_2$ in $\mathcal{P}(M)/{\sim_F}$. From $st \in F$ it follows that $s\{1, e\}t \cap F \neq \emptyset$ and hence $s\{e\}t \cap F \neq \emptyset$. Therefore $set \in F$.

Assume now that the implication (10) holds and towards a contradiction assume that R is a subsemiring of $\mathcal{P}(M)$ with the zero-element $[A]_{\sim_F}$ and the one-element $[B]_{\sim_F}$, where $A \not\sim_F B$. By Lemma 6.7(3), we can choose A and B to be the largest elements in their classes with respect to $\subseteq$. Then we have $A \subseteq B$ because $[A \cup B]_{\sim_F} = [A]_{\sim_F} \cup [B]_{\sim_F} = [B]_{\sim_F}$. Similarly, $A^2 \sim_F A$ implies

$A^2 \subseteq A$ and therefore A contains an idempotent $e \in A$ (take a^ω for any $a \in A$). Finally we have $AB \sim_F A$, which implies $AB \subseteq A$.

Since $A \not\sim_F B$, we can distinguish the following cases:

Case 1. There are $s, t \in M$, $a \in A$ with $sat \in F$ and $sBt \cap F = \emptyset$. Since $a \in A \subseteq B$, this immediately leads to a contradiction.

Case 2. There are $s, t \in M$, $b \in B$ with $sbt \in F$ and $sAt \cap F = \emptyset$. We have $eb \in AB \subseteq A$ and $sebt \in F$ by (10), which again yields a contradiction. □

We do not have a nice characterization for solvability of the multiplicative semigroup of $\mathcal{P}(M)/{\sim_F}$. Let us conclude this section with an application of Theorem 6.9:

Example 6.11. Consider a language of the form $L = \Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots a_k \Sigma^*$ for $a_1, \ldots, a_k \in \Sigma$, which is a so-called *piecewise testable language*. We claim that PCFG-IP(L, Σ) is decidable in NL. First, since $uw \in L$ implies $uvw \in L$ for all $u, v, w \in \Sigma^*$, the syntactic monoid M and the accepting subset $F \subseteq M$ of L clearly satisfy condition (10) from Proposition 6.10. Second, $\mathcal{P}(M)_+$ and hence also $(\mathcal{P}(M)/{\sim_F})_+$ are aperiodic. Third, we show that $\mathcal{P}(M)_\bullet$ and hence also $(\mathcal{P}(M)/{\sim_F})_\bullet$ are aperiodic. Simon's theorem [36] states that a language is piecewise testable if and only if its syntactic monoid is $\mathcal{J}$-trivial.[4] We claim that $\mathcal{P}(M)_\bullet$ is also $\mathcal{J}$-trivial, in particular aperiodic. Let $A, B \in \mathcal{P}(M)$ such that $A \equiv_{\mathcal{J}} B$, i.e., $A = XBY$ and $B = X'AY'$ for some $X, Y, X', Y' \in \mathcal{P}(M)$. Consider the directed bipartite graph on $A \uplus B$ with edges

$$\{(a, b) \in A \times B \mid a \leq_{\mathcal{J}} b\} \cup \{(b, a) \in B \times A \mid b \leq_{\mathcal{J}} a\}.$$

Every vertex has at least one outgoing and one incoming edge, which means that it belongs to a non-trivial strongly connected component. Assume now that $A \neq B$. W.l.o.g. assume that there exists $a \in A$ with $a \notin B$. Since a belongs to a non-trivial strongly connected component, there exists $b \in B$ with $a \equiv_{\mathcal{J}} b$, which contradicts the fact that M is $\mathcal{J}$-trivial. Hence, we have $A = B$.

6.2 Intersection problem for regular expressions

In the following we consider the intersection non-emptiness problem for regular expressions. Given an alphabet Σ and a language $L \subseteq \Sigma^*$, the REG-*intersection non-emptiness problem for L*, denoted by REG-IP(L, Σ), is the following decision problem:

Input A regular expression t over Σ
Question Does $L(t) \cap L \neq \emptyset$ hold?

Lemma 6.12. REG-IP(L, Σ) *is* NC^1*-hard for every non-empty language* $L \subseteq \Sigma^*$.

Proof. We present a simple reduction from EEP$(\mathbb{B}_2)$. Given a Boolean expression t, we replace every occurrence of the truth value 0 (resp., 1) by $\emptyset$ (resp., Σ^*). Every occurrence of the Boolean operator $\vee$ (resp., $\wedge$) is replaced by $\cup$ (resp., $\cdot$). Let t' be the resulting regular expression. If $[t] = 1$ then $L(t') = \Sigma^*$, and if $[t] = 0$ then $L(t') = \emptyset$. Hence $[t] = 1$ if and only if $L(t') \cap L \neq \emptyset$. □

As for PCFG-IP(L, Σ), Lemma 6.12 motivates to consider a restricted problem. A *positive regular expression* is a regular expression which does not use the empty set as an atom. We study the following decision problem PREG-IP(L, Σ):

[4]Here are the relevant definitions concerning $\mathcal{J}$-trivial monoids: On a monoid M one defines the $\mathcal{J}$-preorder $\leq_{\mathcal{J}}$ by $s \leq_{\mathcal{J}} t$ if $s = xty$ for some $x, y \in M$. The corresponding $\mathcal{J}$-equivalence $\equiv_{\mathcal{J}}$ is defined by $s \equiv_{\mathcal{J}} t$ if and only if $s \leq_{\mathcal{J}} t \leq_{\mathcal{J}} s$; it is an equivalence relation. The monoid M is called $\mathcal{J}$-trivial if every equivalence class with respect to $\equiv_{\mathcal{J}}$ contains only one element.

Input A positive regular expression t over Σ
Question Does $L(t) \cap L \neq \emptyset$ hold?

We fix the same terminology as in Section 6.1.1: Let L be a regular language, $h\colon \Sigma^* \to M$ a surjective homomorphism into the syntactic monoid of M of L, and $F \subseteq M$ such that $L = h^{-1}(F)$. Given an algebraic structure $\mathcal{A}$ with domain $A \subseteq 2^M$ we define the decision problem $\mathsf{TEP}(\mathcal{A}, F)$:

Input A tree $(\mathcal{T}, \leq_{\mathcal{T}})$ over $\mathcal{A}$
Question Does $[\mathcal{T}] \cap F \neq \emptyset$ hold?

Proposition 6.13. *$\mathsf{REG\text{-}IP}(L, \Sigma)$ is TC^0-reducible to $\mathsf{TEP}(\mathcal{P}_0(M), F)$.*

Proof. Let t be the given regular expression, which we convert into a tree $\mathcal{T}$ in ancestor representation over the algebra $(2^{\Sigma^*}, \cup, \cdot, *) = (\mathcal{P}_0(\Sigma^*), *)$ (which is also known as a Kleene algebra). In [20] it was shown that a given regular expression can be transformed in TC^0 into an equivalent regular expression of logarithmic depth. Hence, we can assume that $\mathcal{T}$ has logarithmic depth.

Consider the Kleene algebra $(2^M, \cup, \cdot, *) = (\mathcal{P}_0(M), *)$ where X^* is defined as

$$X^* = \bigcup_{i=0}^{\infty} X^i = \bigcup_{i=0}^{|M|} X^i \tag{11}$$

The last equality holds since M is finite. By replacing each input gate $v \to X$ for $X \subseteq \Sigma^*$ by $v \to h(X)$, we get a tree $\mathcal{T}'$ of logarithmic depth over $(\mathcal{P}_0(M), *)$ such that $L(t) \cap L \neq \emptyset$ if and only if $[\mathcal{T}'] \cap F \neq \emptyset$.

To eliminate the $*$-operation, we introduce intermediate gates which compute the partial sums and products in (11). This yields a circuit C of logarithmic depth (note that $|M|$ is a fixed constant). Moreover, the transformation $\mathcal{T}' \mapsto C$ is a guarded transduction, and hence we can compute the EC-representation of the logarithmic depth circuit C in TC^0 by Lemma 2.5. After unfolding the circuit again (which by Lemma 2.2 can be done in TC^0), we obtain a tree over $\mathcal{P}_0(M)$ with the desired property. □

Corollary 6.14. *For every regular language $L \subseteq \Sigma^*$ the problem $\mathsf{REG\text{-}IP}(L, \Sigma)$ is in NC^1.*

Lemma 6.15. *$\mathsf{PREG\text{-}IP}(L, \Sigma)$ is equivalent to $\mathsf{TEP}(\mathcal{P}(M), F)$ with respect to TC^0-reductions.*

Proof. For the reduction from $\mathsf{PREG\text{-}IP}(L, \Sigma)$ to $\mathsf{TEP}(\mathcal{P}(M), F)$, notice that the tree $\mathcal{T}'$ constructed in the proof of Proposition 6.13 is indeed defined over the semiring $\mathcal{P}(M)$ if the input regular expression is positive.

Let us now reduce $\mathsf{TEP}(\mathcal{P}(M), F)$ to $\mathsf{PREG\text{-}IP}(L, \Sigma)$. Let $\mathcal{T} = (V, E, \lambda, v_0)$ be a tree over $\mathcal{P}(M)$, given in ancestor representation. We compute a regular expression (represented as a tree $\mathcal{T}'$ in ancestor representation) over the alphabet Σ by replacing the input gates as follows: If $v \to \{m_1, \dots, m_k\} \in \mathcal{P}(M)$ is an input gate in $\mathcal{T}$, define $v \to \bigcup_{i=1}^k w_i$ in $\mathcal{T}'$ where $w_i \in \Sigma^*$ is any word with $h(w_i) = m_i$. Then, for every node $v \in V$ we have $[v]_{\mathcal{T}} = h(L_v)$, where L_v is the language produced by the regular expression that is rooted in v. In particular, we have $h(L(\mathcal{T}')) = [\mathcal{T}]$. Therefore, $[\mathcal{T}] \cap F \neq \emptyset$ if and only if $L(\mathcal{T}') \cap L \neq \emptyset$. □

Lemma 6.16. *$\mathsf{TEP}(\mathcal{P}(M), F)$ is equivalent to $\mathsf{TEP}(\mathcal{P}(M)/{\sim_F})$ with respect to AC^0-Turing-reductions.*

Proof. Similar to Lemma 6.8. □

From Corollary 1.4, Lemma 6.15 and Lemma 6.16 we obtain:

Theorem 6.17. *$\mathsf{PREG\text{-}IP}(L, \Sigma)$ is equivalent to $\mathsf{TEP}(\mathcal{P}(M)/{\sim_F})$ with respect to TC^0-Turing-reductions. Therefore,*

- $\mathsf{PREG\text{-}IP}(L,\Sigma)$ *is* NC^1*-complete if* $\mathcal{P}(M)/{\sim_F}$ *is not* $\{0,1\}$*-free or its multiplicative semigroup is not solvable,*
- $\mathsf{PREG\text{-}IP}(L,\Sigma)$ *is in* TC^0 *if* $\mathcal{P}(M)/{\sim_F}$ *is* $\{0,1\}$*-free and its multiplicative semigroup is solvable.*

7 CONCLUSION AND OUTLOOK

We proved a dichotomy result for the circuit evaluation problem for finite semirings: If (i) the semiring has no subsemiring with an additive and multiplicative identity and both are different and (ii) the multiplicative subsemigroup is solvable, then the circuit evaluation problem is in $\mathsf{DET} \subseteq \mathsf{NC}^2$, otherwise it is P-complete.

The ultimate goal would be to obtain such a dichotomy for all finite algebraic structures. One might ask whether for every finite algebraic structure $\mathcal{A}$, $\mathsf{CEP}(\mathcal{A})$ is P-complete or in NC. It is known that under the assumption $\mathsf{P} \neq \mathsf{NC}$ there exist problems in $\mathsf{P} \setminus \mathsf{NC}$ that are not P-complete [39]. In [11] it is shown that every circuit evaluation problem $\mathsf{CEP}(\mathcal{A})$ is equivalent to a circuit evaluation problem $\mathsf{CEP}(A, \circ)$, where $\circ$ is a binary operation.

ACKNOWLEDGMENTS

We thank Benjamin Steinberg for helpful discussions on semigroups. We are grateful to Volker Diekert for pointing out to us the proof of the implication ($3 \Rightarrow 4$) in the proof of Lemma 3.1. We thank the anonymous reviewers for helpful comments.

The third and fourth author were supported by the DFG research project Lo 748/12-1.

REFERENCES

[1] Manindra Agrawal and Somenath Biswas. Primality and identity testing via chinese remaindering. *Journal of the Association for Computing Machinery*, 50(4):429–443, 2003.
[2] Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM Journal on Computing*, 38(5):1987–2006, 2009.
[3] Eric Allender, Jia Jiao, Meena Mahajan, and V. Vinay. Non-commutative arithmetic circuits: Depth reduction and size lower bounds. *Theor. Comput. Sci.*, 209(1-2):47–86, 1998.
[4] Jorge Almeida, Stuart Margolis, Benjamin Steinberg, and Mikhail Volkov. Representation theory of finite semigroups, semigroup radicals and formal language theory. *Transactions of the American Mathematical Society*, 361(3):1429–1461, 2009.
[5] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.
[6] Karl Auinger and Benjamin Steinberg. Constructing divisions into power groups. *Theoretical Computer Science*, 341(1–3):1–21, 2005.
[7] D. A. M. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38:150–164, 1989.
[8] D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.
[9] David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 35(4):941–952, 1988.
[10] Martin Beaudry and Markus Holzer. The complexity of tensor circuit evaluation. *Computational Complexity*, 16(1):60–111, 2007.
[11] Martin Beaudry and Pierre McKenzie. Circuits, matrices, and nonassociative computation. *Journal of Computer and System Sciences*, 50(3):441–455, 1995.
[12] Martin Beaudry, Pierre McKenzie, Pierre Péladeau, and Denis Thérien. Finite monoids: From word to circuit evaluation. *SIAM Journal on Computing*, 26(1):138–152, 1997.
[13] Samuel R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proceedings of the 19th Annual Symposium on Theory of Computing (STOC 87)*, pages 123–131. ACM Press, 1987.
[14] Ashok K. Chandra, Steven Fortune, and Richard J. Lipton. Unbounded fan-in circuits and associative functions. *J. Comput. Syst. Sci.*, 30(2):222–234, 1985.
[15] Ashok K. Chandra, Larry J. Stockmeyer, and Uzi Vishkin. Constant depth reducibility. *SIAM J. Comput.*, 13(2):423–439, 1984.
[16] Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[17] Stephen A. Cook and Lila Fontes. Formal theories for linear algebra. *Logical Methods in Computer Science*, 8(1), 2012.

[18] M. Elberfeld, A. Jakoby, and T. Tantau. Algorithmic meta theorems for circuit classes of constant and logarithmic depth. In *Proceedings of STACS 2012*, volume 14 of *LIPIcs*, pages 66–77. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012.

[19] Kousha Etessami. Counting quantifiers, successor relations, and logarithmic space. *Journal of Computer and System Sciences*, 54(3):400 – 411, 1997.

[20] Moses Ganardi and Markus Lohrey. A universal tree balancing theorem. *CoRR*, abs/1704.08705, 2017.

[21] Jonathan S. Golan. *Semirings and their Applications.* Springer, 1999.

[22] Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory.* Oxford University Press, 1995.

[23] Oscar H. Ibarra and Shlomo Moran. Probabilistic algorithms for deciding equivalence of straight-line programs. *Journal of the ACM*, 30(1):217–228, 1983.

[24] Neil D. Jones and William T. Laaser. Complete problems for deterministic polynomial time. *Theor. Comput. Sci.*, 3(1):105–117, 1976.

[25] Daniel König and Markus Lohrey. Evaluation of circuits over nilpotent and polycyclic groups. *Algorithmica*, 2017.

[26] S. Rao Kosaraju. On parallel evaluation of classes of circuits. In *Proceedings of the 10th Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 1990*, volume 472 of *Lecture Notes in Computer Science*, pages 232–237. Springer, 1990.

[27] Richard E. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7(1):18–20, 1975.

[28] Markus Lohrey. On the parallel complexity of tree automata. In Aart Middeldorp, editor, *Proceedings of the 12th International Conference on Rewrite Techniques and Applications (RTA 2001), Utrecht (The Netherlands)*, volume 2051 of *Lecrture Notes in Computer Science*, pages 201–215. Springer, 2001.

[29] Pierre McKenzie and Klaus W. Wagner. The complexity of membership problems for circuits over sets of natural numbers. *Computational Complexity*, 16(3):211–244, 2007.

[30] Gary L. Miller, Vijaya Ramachandran, and Erich Kaltofen. Efficient parallel evaluation of straight-line code and arithmetic circuits. *SIAM J. Comput.*, 17(4):687–695, 1988.

[31] Gary L. Miller and Shang-Hua Teng. The dynamic parallel complexity of computational circuits. *SIAM J. Comput.*, 28(5):1664–1688, 1999.

[32] Cristopher Moore, Denis Thérien, François Lemieux, Joshua Berman, and Arthur Drisko. Circuits and expressions with nonassociative gates. *J. Comput. Syst. Sci.*, 60(2):368–394, 2000.

[33] John Rhodes and Benjamin Steinberg. *The q-theory of Finite Semigroups.* Springer, 2008.

[34] Alexander A. Rubtsov and Mikhail N. Vyalyi. Regular realizability problems and context-free languages. In *Proceedings of the 17th International Workshop on Descriptional Complexity of Formal Systems, DCFS 2015*, volume 9118 of *Lecture Notes in Computer Science*, pages 256–267. Springer, 2015.

[35] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

[36] Imre Simon. Piecewise testable events. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages, 1975*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer, 1975.

[37] Stephen D. Travers. The complexity of membership problems for circuits over sets of integers. *Theor. Comput. Sci.*, 369(1-3):211–229, 2006.

[38] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644, 1983.

[39] Heribert Vollmer. The gap-language-technique revisited. In *Proceedings of the 4th Workshop on Computer Science Logic, CSL '90*, volume 533 of *Lecture Notes in Computer Science*, pages 389–399. Springer, 1990.

[40] Heribert Vollmer. *Introduction to Circuit Complexity.* Springer, 1999.

Received January 2018; revised January 2018; accepted January 2018