

# A Model-Theoretic Characterization of Constant-Depth Arithmetic Circuits

Anselm Haak and Heribert Vollmer<sup>(✉)</sup>

Institut für Theoretische Informatik, Leibniz Universität Hannover,  
Hannover, Germany  
{haak,vollmer}@thi.uni-hannover.de

**Abstract.** We study the class  $\#AC^0$  of functions computed by constant-depth polynomial-size arithmetic circuits of unbounded fan-in addition and multiplication gates. No model-theoretic characterization for arithmetic circuit classes is known so far. Inspired by Immerman’s characterization of the Boolean class  $AC^0$ , we remedy this situation and develop such a characterization of  $\#AC^0$ . Our characterization can be interpreted as follows: Functions in  $\#AC^0$  are exactly those functions counting winning strategies in first-order model checking games. A consequence of our results is a new model-theoretic characterization of  $TC^0$ , the class of languages accepted by constant-depth polynomial-size majority circuits.

## 1 Introduction

Going back to questions posed by Heinrich Scholz and Günter Asser in the early 1960s, Ronald Fagin [6] laid the foundations for the areas of finite model theory and descriptive complexity theory. He characterized the complexity class NP as the class of those languages that can be defined in predicate logic by existential second-order sentences:  $NP = ESO$ . His result is the cornerstone of a wealth of further characterizations of complexity classes, cf. the monographs [5, 11, 12].

Fagin’s Theorem has found a nice generalization: Considering first-order formulae with a free relational variable, instead of asking if there *exists* an assignment to this variable that makes the formula true (ESO), we now ask to *count* how many assignments there are. In this way, the class  $\#P$  is characterized:  $\#P = \#FO$  [13].

But also “lower” complexity classes, defined by families of Boolean circuits, have been considered in a model-theoretical way. Most important for us is the characterization of the class  $AC^0$ , the class of languages accepted by families of Boolean circuits of unbounded fan-in, polynomial size and constant depth, by first-order formulae. This correspondence goes back to Immerman and his co-authors [2, 10], but was somewhat anticipated by [8]. Informally, this may be written as  $AC^0 = FO$ ; and there are two ways to make this formally correct—a non-uniform one:  $AC^0 = FO[Arb]$ , and a uniform one:  $FO\text{-uniform } AC^0 = FO[+, \times]$  (for details, see below).

---

Supported by DFG grant VO 630/8-1.

In the same way as  $\#P$  can be seen as the counting version of  $NP$ , there is a counting version of  $AC^0$ , namely  $\#AC^0$ , the class of those functions counting accepting proof-trees of  $AC^0$ -circuits. A proof-tree is a minimal sub-circuit of the original circuit witnessing that it outputs 1. Equivalently,  $\#AC^0$  can be characterized as those functions computable by polynomial-size constant-depth circuits with unbounded fan-in  $+$  and  $\times$  gates (and Boolean inputs); for this reason we also speak of *arithmetic* circuit classes.

For such arithmetic classes, no model-theoretic characterization is known so far. Our *rationale* is as follows: A Boolean circuit accepts its input if it has at least one proof-tree. An FO-formula (w.l.o.g. in prenex normal form) holds for a given input if there are Skolem functions determining values for the existentially quantified variables, depending on those variables quantified to the left. By establishing a one-one correspondence between proof-trees and Skolem functions, we show that the class  $\#AC^0$ , defined by counting proof-trees, is equal to the class of functions counting Skolem functions, or, alternatively, winning-strategies in first-order model-checking games:  $AC^0 = \#Skolem-FO = \#Win-FO$ . We prove that this equality holds in the non-uniform as well as in the uniform setting.

It seems a natural next step to allow first-order formulae to “talk” about winning strategies, i.e., allow access to  $\#Win-FO$ -functions (like to an oracle). We will prove that in doing so, we obtain a new model-theoretic characterization of the circuit class  $TC^0$  of polynomial-size constant-depth MAJORITY circuits.

This paper is organized as follows: In the upcoming section, we will introduce the relevant circuit classes and logics, and we state characterizations of the former by the latter known from the literature. We will also recall arithmetic circuit classes and define our logical counting classes  $\#Skolem-FO$  and  $\#Win-FO$ . Section 3 proves our characterization of non-uniform  $\#AC^0$ , while Sect. 4 proves our characterization of uniform  $\#AC^0$ . Section 5 presents our new characterization of the circuit class  $TC^0$ . Finally, Sect. 6 concludes with some open questions.

Due to space restrictions, many proofs have to be omitted and will be given in the full paper.

## 2 Circuit Classes, Counting Classes, and Logic

### 2.1 Non-uniform Circuit Classes

A relational vocabulary is a tuple  $\sigma = (R_1^{a_1}, \dots, R_k^{a_k})$ , where  $R_i$  are relation symbols and  $a_i$  their arities,  $1 \leq i \leq k$ . We define first-order formulae over  $\sigma$  as usual (see, e.g., [5, 11]). First-order structures fix the set of elements (the universe) as well as interpretations for the relation symbols in the vocabulary. Semantics is defined as usual. For a structure  $\mathcal{A}$ ,  $|\mathcal{A}|$  denotes its universe. We only consider finite structures here, which means their universes are finite.

Since we want to talk about languages accepted by Boolean circuits, we will use the *vocabulary*

$$\tau_{\text{string}} = (\leq^2, S^1)$$

of *binary strings*. A binary string is represented as a structure over this vocabulary as follows: Let  $w \in \{0, 1\}^*$  with  $|w| = n$ . Then the structure representing this string has universe  $\{0, \dots, n-1\}$ ,  $\leq^2$  is interpreted as the  $\leq$ -relation on the natural numbers and  $x \in S$ , iff the  $x$ 'th bit of  $w$  is 1. The structure corresponding to string  $w$  will be called  $\mathcal{A}_w$ . Vice versa, structure  $\mathcal{A}_w$  is simply encoded by  $w$  itself: The bits define which elements are in the  $S$ -relation—the universe and the order are implicit. This encoding can be generalized to binary encodings of arbitrary  $\sigma$ -structures  $\mathcal{A}$ . We will use the notation  $\text{enc}_\sigma(\mathcal{A})$  for such an encoding.

A Boolean circuit  $C$  is a directed acyclic graph (dag), whose nodes (also called gates) are marked with either a Boolean function (in our case  $\wedge$  or  $\vee$ ), a constant (0 or 1), or a (possibly negated) query of a particular position of the input. Also, one gate is marked as the output gate. On any input  $x$ , a circuit computes a Boolean function  $f_C$  by evaluating all gates according to what they are marked with. The value of the output gate gives then the result of the computation of  $C$  on  $x$ , i.e.,  $f_C(x)$ .

A single circuit computes only a finite Boolean function. When we want circuits to work on different input lengths, we have to consider families of circuits: A family contains one circuit for any input length  $n \in \mathbb{N}$ . Families of circuits allow us to talk about languages being accepted by circuits: A circuit family  $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$  is said to accept (or decide) the language  $L$ , if it computes its characteristic function  $c_L$ :

$$C_{|x|}(x) = c_L(x) \text{ for all } x.$$

Since we will describe Boolean circuits by FO-formulae, we define the vocabulary

$$\tau_{\text{circ}} = (E^2, G_\wedge^1, G_\vee^1, B^1, r^1),$$

the *vocabulary of Boolean circuits*. The relations are interpreted as follows:

- $E(x, y)$ :  $y$  is a child of  $x$
- $G_\wedge(x)$ : gate  $x$  is an and-gate
- $G_\vee(x)$ : gate  $x$  is an or-gate
- $B(x)$ : Gate  $x$  is a true leaf of the circuit
- $r(x)$ :  $x$  is the root of the circuit

The definition from [11] is more general because it allows negations to occur arbitrary in a circuit. Here we only consider circuits in *negation normal form*, i.e., negations are only applied to input bits. This restriction is customary for arithmetic circuits like for the class  $\#\text{AC}^0$  to be defined below.

The complexity classes in circuit complexity are classes of languages that can be decided by circuit families with certain restrictions on their depth or size. The depth here is the length of a longest path from any input gate to the output gate of a circuit and the size is the number of non-input gates in a circuit. Depth and size of a circuit family are defined as functions accordingly.

**Definition 1.** The class  $\text{AC}^0$  is the class of all languages decidable by Boolean circuit families of constant depth and polynomial size.

In this definition we do not have any restrictions on the computability of the function  $n \mapsto \langle C_n \rangle$ , i.e., the function computing (an encoding of) the circuit for a given input length. This phenomenon is referred to as *non-uniformity*, and it leads to undecidable problems in  $\text{AC}^0$ . In first-order logic there is a class that has a similar concept, the class  $\text{FO}[\text{Arb}]$ , to be defined next.

For arbitrary vocabularies  $\tau$ , we consider formulae over  $\tau_{\text{string}} \cup \tau$  and our input structures will always be  $\tau_{\text{string}}$ -structures  $\mathcal{A}_w$  for a string  $w \in \{0, 1\}^*$ . To evaluate a formula we additionally specify a (non-uniform) family  $I = (I_n)_{n \in \mathbb{N}}$  of interpretations of the relation symbols in  $\tau$ . For  $\mathcal{A}_w$  and  $I$  as above we now evaluate  $\mathcal{A}_w \models_I \varphi$  by using the universe of  $\mathcal{A}_w$  and the interpretations from both  $\mathcal{A}_w$  and  $I_{|w|}$ . The language defined by a formula  $\varphi$  and a family of interpretations  $I$  is

$$L_I(\varphi) =_{\text{def}} \{w \in \{0, 1\}^* \mid \mathcal{A}_w \models_I \varphi\}$$

This leads to the following definition of  $\text{FO}[\text{Arb}]$  (equivalent to the one given in [14]):

**Definition 2.** A language  $L$  is in  $\text{FO}[\text{Arb}]$ , if there are an arbitrary vocabulary  $\tau$ , a first-order sentence  $\varphi$  over  $\tau_{\text{string}} \cup \tau$  and a family  $I = (I_n)_{n \in \mathbb{N}}$  of interpretations of the relation symbols in  $\tau$  such that:

$$L_I(\varphi) = L$$

It is known that the circuit complexity class  $\text{AC}^0$  and the model theoretic class  $\text{FO}[\text{Arb}]$  are in fact the same:

**Theorem 3** (see, e.g., [14]).  $\text{AC}^0 = \text{FO}[\text{Arb}]$ .

## 2.2 Uniform Circuit Classes

As already stated, non-uniform circuits are able to solve undecidable problems, even when restricting size and depth of the circuits dramatically. Thus, the non-uniformity somewhat obscures the real complexity of problems. There are different notions of uniformity to deal with this problem: The computation of the circuit  $C_{|x|}$  from  $x$  must be possible within certain bounds, e.g. polynomial time, logarithmic space, logarithmic time. Since we are dealing with FO-formulae, the type of uniformity we will need is first-order uniformity, to be defined in this section.

In the logical languages, “uniformity” means we now remove the non-uniform family of interpretations from the definition of  $\text{FO}[\text{Arb}]$ , and replace it with two special symbols for arithmetic, a 3-ary relation  $+$  (with the intended interpretation  $+(i, j, k)$  iff  $i + j = k$ ) and a 3-ary relation  $\times$  (with the intended interpretation  $\times(i, j, k)$  iff  $i \cdot j = k$ ).

**Definition 4.** A language  $L$  is in  $\text{FO}[+, \times]$ , if there is a first-order sentence  $\varphi$  over  $\tau_{\text{string}} \cup \{+, \times\}$  such that

$$\mathcal{A}_w \models_I \varphi \Leftrightarrow w \in L,$$

where  $I$  interprets  $+$  and  $\times$  in the intended way.

In the circuit world, as mentioned, “uniformity” means we can access from any given input structure  $\mathcal{A}_w$  also the circuit  $C_{|w|}$ . The way we achieve this is via *FO-interpretations*.

In the following, for any vocabulary  $\sigma$ ,  $\text{STRUC}[\sigma]$  denotes the set of all structures over  $\sigma$ .

**Definition 5.** Let  $\sigma, \tau$  be vocabularies,  $\tau = (R_1^{a_1}, \dots, R_r^{a_r})$ , and let  $k \in \mathbb{N}$ . A *first-order interpretation* (or *FO-interpretation*)

$$I: \text{STRUC}[\sigma] \rightarrow \text{STRUC}[\tau]$$

is given by a tuple of FO-formulae  $\varphi_0, \varphi_1, \dots, \varphi_r$  over the vocabulary  $\sigma$ .  $\varphi_0$  has  $k$  free variables and  $\varphi_i$  has  $k \cdot a_i$  free variables for all  $i \geq 1$ . For each structure  $\mathcal{A} \in \text{STRUC}[\sigma]$ , these formulae define the structure

$$I(\mathcal{A}) = (|I(\mathcal{A})|, R_1^{I(\mathcal{A})}, \dots, R_r^{I(\mathcal{A})}) \in \text{STRUC}[\tau],$$

where the universe is defined by  $\varphi_0$  and the relations are defined by  $\varphi_1, \dots, \varphi_r$  in the following way:

$$|I(\mathcal{A})| = \{\langle b^1, \dots, b^k \rangle \mid \mathcal{A} \models \varphi_0(b^1, \dots, b^k)\} \text{ and}$$

$$R_i^{I(\mathcal{A})} = \{(\langle b_1^1, \dots, b_1^k \rangle, \dots, \langle b_{a_i}^1, \dots, b_{a_i}^k \rangle) \in |I(\mathcal{A})|^{a_i} \mid \mathcal{A} \models \varphi_i(b_1^1, \dots, b_{a_i}^k)\}$$

The name FO-interpretations was used, e.g., in [4]. Sometimes they are also referred to as first-order queries, see, e.g., [11]. They are not to be confused with interpretations of relation symbols as in Sect. 2.1. It is customary to use the same symbol  $I$  in both cases.

Analogously,  $\text{FO}[+, \times]$ -interpretations are interpretations given by tuples of  $\text{FO}[+, \times]$ -formulae.

**Definition 6.** A circuit family  $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$  is said to be  $\text{FO}[+, \times]$ -uniform if there is an  $\text{FO}[+, \times]$ -interpretation

$$I: \text{STRUC}[\tau_{\text{string}}] \rightarrow \text{STRUC}[\tau_{\text{circuit}}]$$

mapping from an input word  $w$  given as a structure  $\mathcal{A}_w$  over  $\tau_{\text{string}}$  to the circuit  $C_{|w|}$  given as a structure over the vocabulary  $\tau_{\text{circuit}}$ .

Now we can define the FO-uniform version of  $\text{AC}^0$ :

**Definition 7.**  $\text{FO}[+, \times]$ -uniform  $\text{AC}^0$  is the class of all languages that can be decided by  $\text{FO}[+, \times]$ -uniform  $\text{AC}^0$  circuit families.

Thus, if  $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$  is an  $\text{FO}[+, \times]$ -uniform circuit family, we can define from any given input structure  $\mathcal{A}_w$  also the circuit  $C_{|w|}$  in a first-order way.

Interestingly, uniform  $\text{AC}^0$  coincides with FO with built-in arithmetic:

**Theorem 8** (see, e.g., [11]).  $\text{FO}[+, \times]$ -uniform  $\text{AC}^0 = \text{FO}[+, \times]$ .

Alternatively, we can replace  $+$  and  $\times$  in the above theorem by the binary symbol BIT with the meaning  $\text{BIT}(i, j)$  iff the  $i$ th bit in the binary representation of  $j$  is 1, see also [11].

### 2.3 Counting Classes

Building on the previous definitions we want to define next counting classes. The objects counted on circuits are proof trees: A *proof tree* is a minimal subtree showing that a circuit evaluates to true for a given input. For this, we first unfold the given circuit into tree shape, and we further require that it is in negation normal form. A proof tree then is a tree we get by choosing for any  $\vee$ -gate exactly one child and for any  $\wedge$ -gate all children, such that every leaf which we reach in this way is a true literal.

Now,  $\#AC^0$  is the class of functions that “count proof trees of  $AC^0$  circuits”:

**Definition 9.** ( $FO[+, \times]$ -uniform)  $\#AC^0$  is the class of all functions  $f: \{0, 1\}^* \rightarrow \mathbb{N}$  for which there is a ( $FO[+, \times]$ -uniform) circuit family  $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$  such that for any word  $x$ ,  $f(x)$  equals the number of proof trees of  $C_{|x|}(x)$ .

It is the aim of this paper to give model-theoretic characterizations of these classes. The only model-theoretic characterization of a counting class that we are aware of is the following: In [13], a counting version of FO was defined, inspired by Fagin’s characterization of NP: Functions in this class count assignments to free relational variables in FO-formulas. However, it is known that  $\#P = \#FO$ , i.e., this counting version of FO coincides with the much higher counting class  $\#P$  of functions counting accepting paths of nondeterministic polynomial-time Turing machines. It is known that  $\#AC^0 \subsetneq \#P$ . Thus, we need some weaker notion of counting.

Suppose we are given a  $\tau_{\text{string}}$ -formula  $\varphi$  in prenex normal form,

$$\varphi \triangleq \exists y_1 \forall z_1 \exists y_2 \forall z_2 \dots \exists y_{k-1} \forall z_{k-1} \exists y_k \psi(\bar{y}, \bar{z})$$

for quantifier-free  $\psi$ . If we want to satisfy  $\varphi$  in a word model  $\mathcal{A}_w$ , we have to find an assignment for  $y_1$  such that for all  $z_1$  we have to find an assignment for  $y_2 \dots$  such that  $\psi$  with the chosen variables holds in  $\mathcal{A}_w$ . Thus, the number of ways to satisfy  $\varphi$  consists in the number of picking the suitable  $y_i$ , depending on the universally quantified variables to the left, such that  $\psi$  holds, in other words, the number of Skolem functions for the existentially quantified variables.

**Definition 10.** A function  $g: \{0, 1\}^* \rightarrow \mathbb{N}$  is in the class  $\#Skolem-FO[Arb]$  if there is a vocabulary  $\tau$ , a sequence of interpretations  $I = (I_n)_{n \in \mathbb{N}}$  for  $\tau$  and a first-order sentence  $\varphi$  over  $\tau_{\text{string}} \cup \tau$  in prenex normal form

$$\varphi \triangleq \exists y_1 \forall z_1 \exists y_2 \forall z_2 \dots \exists y_{k-1} \forall z_{k-1} \exists y_k \psi(\bar{y}, \bar{z})$$

such that for all  $w \in \{0, 1\}^*$ ,  $g(w)$  is equal to the number of tuples  $(f_1, \dots, f_k)$  of functions such that

$$\mathcal{A}_w \models_I \forall z_1 \dots \forall z_{k-1} \psi(f_1, f_2(z_1), \dots, f_k(z_1, \dots, z_{k-1}), z_1, \dots, z_{k-1})\}$$

This means that  $\#Skolem-FO[Arb]$  contains those functions that, for a fixed FO-formula, map an input  $w$  to the number of Skolem functions on  $\mathcal{A}_w$ .

A different view on this counting class is obtained by recalling the well-known game-theoretic approach to first-order model checking: Model checking for FO-formulae (in prenex normal form) can be characterized using a two player game: The verifier wants to show that the formula evaluates to true, whereas the falsifier wants to show that it does not. For each quantifier, one of the players chooses an action: For an existential quantifier, the verifier chooses which element to take (because he needs to prove that there is an element). For a universal quantifier, the falsifier chooses which element to take (because he needs to prove that there is a choice falsifying the formula following after the quantifier). When all quantifiers have been addressed, it is checked whether the quantifier-free part of the formula is true or false. If it is true, the verifier wins. Else, the falsifier wins. Now the formula is fulfilled by a given model, iff there is a winning strategy (for the verifier).

**Definition 11.** A function  $f$  is in  $\#Win-FO[Arb]$ , if there are a vocabulary  $\tau$ , a sequence of interpretations  $I = (I_n)_{n \in \mathbb{N}}$  for  $\tau$  and a first-order sentence  $\varphi$  in prenex normal form over  $\tau_{string} \cup \tau$  such that for all  $w \in \{0, 1\}^*$ ,  $f(w)$  equals the number of winning strategies for the verifier in the game for  $\mathcal{A}_w \models_I \varphi$ .

The correspondence between Skolem functions and winning strategies has been observed in far more general context, see, e.g., [7]. In our case, this means that

$$\#Skolem-FO[Arb] = \#Win-FO[Arb].$$

Analogously we define the uniform version (where we only state using the notion of the model checking games):

**Definition 12.** A function  $f$  is in  $\#Win-FO[+, \times]$ , if there is a first-order sentence  $\varphi$  in prenex normal form over  $\tau_{string} \cup \{+, \times\}$  such that for all  $w \in \{0, 1\}^*$ ,  $f(w)$  equals the number of winning strategies for the verifier in the game for  $\mathcal{A}_w \models_I \varphi$ , where  $I$  interprets  $+$  and  $\times$  in the intended way.

We will use  $\#Win(\varphi, \mathcal{A}, I)$  ( $\#Win(\varphi, \mathcal{A})$ , resp.) to denote the number of winning strategies for  $\varphi$  evaluated on the structure  $\mathcal{A}$  and the interpretation  $I$  (the structure  $\mathcal{A}$  and the intended interpretation of  $+$  and  $\times$ , resp.).

In the previous two definitions we could again have replaced  $+$  and  $\times$  by BIT.

In the main result of this paper, we will show that the thus defined logical counting classes equal the previously defined counting classes for constant-depth circuits.

### 3 A Model-Theoretic Characterization of $\#AC^0$

We first note that there is a sort of a closed formula for the number of winning strategies of FO-formulae on given input structures:

**Lemma 13.** *Let  $\tau_1, \tau_2$  be vocabularies and  $I$  an interpretation of  $\tau_2$ . Let  $\varphi$  be an FO-formula in prenex normal form over the vocabulary  $\tau_1 \cup \tau_2$  of the form*

$$\varphi \triangleq Q_1 x_1 \dots Q_n x_n \psi,$$

where  $Q_i \in \{\exists, \forall\}$ .

Let  $\mathcal{A}$  be a  $\tau_1$ -structure and  $I$  a sequence of interpretations for  $\tau_2$ . Then the number of winning strategies of  $\mathcal{A} \models_I \varphi$  is the following:

$$\#\text{Win}(\varphi, \mathcal{A}, I) = \Delta_1 \Delta_2 \dots \Delta_n([\mathcal{A} \models_I \varphi(a_1 \dots a_n)]),$$

where

$$\Delta_i = \begin{cases} \sum_{a_i \in |\mathcal{A}|} & , \text{ if } Q_i = \exists \\ \prod_{a_i \in |\mathcal{A}|} & , \text{ if } Q_i = \forall \end{cases}$$

and  $[\mathcal{A} \models_I \varphi(a_1, \dots, a_n)]$  is interpreted as either 0 or 1 depending on its truth value.

In the uniform case,  $\#\text{Win}(\varphi, \mathcal{A})$  is the special case of  $\#\text{Win}(\varphi, \mathcal{A}, I)$  where  $I$  interprets  $+$  and  $\times$  in the intended way.

Our main theorem can now be stated as follows:

**Theorem 14.**  $\#\text{AC}^0 = \#\text{Win-FO}[\text{Arb}]$

The rest of this section is devoted to a proof of this theorem.

*Proof.*  $\subseteq$ : Let  $f$  be a function in  $\#\text{AC}^0$  and  $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$  a  $\text{AC}^0$ -circuit witnessing this. Assume that all  $C_n$  are already trees and all leaves have the same depth (the latter can be achieved easily by adding and-gates with only one input). Also, we can assume that the circuit always uses and- and or-gates alternating beginning with an and-gate in the root. This can be achieved by doubling the depth of the circuit: For every layer of the old circuit we use an and-gate followed by an or-gate. If the original gate in that layer was an and-gate, we just put an (one-input) or-gate on top of every child and connect those or-gates to the and-gate. If the original gate in that layer was an or-gate, we put an and-gate above it with the or-gate as its only child.

Let  $w \in \{0, 1\}^*$  be an input,  $r$  be the root of  $C_{|w|}$  and  $k$  the depth of  $C_n$  for all  $n$ . The value  $f(w)$  can be given as follows:

$$f(w) = \prod_{\substack{y_1 \text{ is a} \\ \text{child of } r}} \sum_{\substack{y_2 \text{ is a} \\ \text{child of } y_1}} \dots \bigcirc_{\substack{y_k \text{ is a} \\ \text{child of } y_{k-1}}} \begin{cases} 1 & , \text{ if } y_k \text{ is a true literal} \\ 0 & , \text{ if } y_k \text{ is a false literal} \end{cases},$$

where  $\bigcirc = \begin{cases} \prod & , \text{ if } k \text{ is odd,} \\ \sum & , \text{ if } k \text{ is even.} \end{cases}$

We will now build an FO-sentence  $\varphi$  over  $\tau_{\text{string}} \cup \tau_{\text{circ}}$  such that for any input  $w \in \{0, 1\}^*$ , the number of winning strategies to verify  $\mathcal{A}_w \models_{\mathcal{C}} \varphi$  equals the number of proof trees of the circuit  $C_{|w|}$  on input  $w$ . Note that the circuit family  $\mathcal{C}$  as a  $\tau_{\text{circ}}$ -structure can directly be used as the non-uniform family of interpretations



for the evaluation of  $\varphi$ . Since only one universe is used for evaluation and it is determined by the input structure  $\mathcal{A}_w$ , the gates in this  $\tau_{\text{circ}}$ -structure have to be tuples of variables ranging over the universe of  $\mathcal{A}_w$ . To simplify the presentation, we assume in the following that we do not need tuples—a single element of the universe already corresponds to a gate. The proof can be generalized to the case where this assumption is dropped.

The sentence  $\varphi$  over  $\tau_{\text{string}} \cup \tau_{\text{circ}}$  can be given as

$$\begin{aligned} \varphi := & \exists y_0 \forall y_1 \exists y_2 \dots Q_k y_k \\ & r(y_0) \wedge \left( \left( \left( \bigwedge_{1 \leq i \leq k} E(y_i, y_{i-1}) \right) \wedge B(y_k) \right) \right. \\ & \left. \vee \bigvee_{\substack{1 \leq i \leq k, \\ i \text{ odd}}} \left( \bigwedge_{1 \leq j < i} (E(y_j, y_{j-1})) \wedge \neg E(y_i, y_{i-1}) \wedge \bigwedge_{i < j \leq k} y_j = r \right) \right), \end{aligned}$$

where  $Q_k = \begin{cases} \exists & , \text{ if } k \text{ is odd,} \\ \forall & , \text{ if } k \text{ is even.} \end{cases}$

We now need to show that the number of winning strategies for  $\mathcal{A}_w \models_I \varphi$  is equal to the number of proof trees of the circuit  $C_{|w|}$  on input  $w$ . For this, let

$$\begin{aligned} \varphi^{(n)}(y_1, \dots, y_n) := & Q_{n+1} y_{n+1} \dots Q_k y_k \\ & \left( \bigwedge_{1 \leq i \leq k} (E(y_i, y_{i-1})) \wedge B(y_k) \right) \vee \\ & \bigvee_{\substack{n+1 \leq i \leq k, \\ i \text{ odd}}} \left( \bigwedge_{1 \leq j < i} (E(y_j, y_{j-1})) \wedge \neg E(y_i, y_{i-1}) \wedge \right. \\ & \left. \bigwedge_{i < j \leq k} y_j = r \right), \end{aligned}$$

where  $Q_{n+1}, \dots, Q_{k-1}$  are the quantifiers preceding  $Q_k$ . Note that the start of the index  $i$  on the big “or” changed compared to  $\varphi$ . Also,  $r$  is notation for the root of the circuit, although we formally do not use constants. In the following we will use the abbreviation

$$\#w(\varphi) = \#\text{Win}(\varphi, \mathcal{A}_w, I).$$

We now show by induction that:

$$\#w(\varphi) = \prod_{\substack{y_1 \text{ is a} \\ \text{child of } r}} \sum_{\substack{y_2 \text{ is a} \\ \text{child of } y_1}} \dots \bigcirc_{\substack{y_n \text{ is a} \\ \text{child of } y_{n-1}}} \#w(\varphi^{(n)}[y_0/r]).$$

Replacing  $y_0$  by  $r$  is only done for simplicity.

Induction basis ( $n = 0$ ): The induction hypothesis here simply states

$$\#w(\varphi) = \#w(\varphi^{(0)}[y_0/r]),$$

which holds by definition.

Induction step ( $n \rightarrow n+1$ ): We can directly use the induction hypothesis here:

$$\#w(\varphi) = \prod_{\substack{y_1 \text{ is a} \\ \text{child of } r}} \sum_{\substack{y_2 \text{ is a} \\ \text{child of } y_1}} \dots \bigcirc_{\substack{y_n \text{ is a} \\ \text{child of } y_{n-1}}} \#w(\varphi^{(n)}[y_0/r]),$$

so it remains to show that

$$\#w(\varphi^{(n)}[y_0/r]) = \bigcirc_{\substack{y_{n+1} \text{ is a} \\ \text{child of } y_n}} \#w(\varphi^{(n+1)}[y_0/r])$$

We distinguish two cases: Depending on whether  $n+1$  is even or odd, the  $(n+1)$ -st quantifier is either an existential or a universal quantifier. In the same way all gates of that depth in the circuits from  $\mathcal{C}$  are either or- or and-gates.

*Case 1:*  $n+1$  is odd, so the  $(n+1)$ -st quantifier is a universal quantifier. Thus, from  $\#w(\varphi^{(n)}[y_0/r])$  we get a  $\prod$ -operator, which is the same we get for an and-gate in the corresponding circuit. We now need to check over which values of  $y_{n+1}$  the product runs:

The big conjunction may only be true if  $y_{n+1}$  is a child of  $y_n$ .

The big disjunction may become true for values of  $y_{n+1}$  which are no children of  $y_n$  only if all variables quantified after  $y_{n+1}$  are set to  $r$  (the choice of  $r$  here is arbitrary and was only made because  $r$  is the only constant in the circuit). Also, the disjunct for  $i = n$  can only be made true if  $y_{n+1}$  is not a child of  $y_n$ , so we can drop it if  $y_{n+1}$  is a child of  $y_n$ . Since for all values of  $y_{n+1}$  that are not children of  $y_n$  we fix all variables quantified afterwards, we get:

$$\begin{aligned} \prod_{y_{n+1} \in |\mathcal{A}_w|} \#w(\varphi^{(n+1)}[y_0/r]) &= \prod_{\substack{y_{n+1} \in |\mathcal{A}_w|, \\ y_{n+1} \text{ is a child of } y_n}} \#w(\varphi^{(n+1)}[y_0/r]) \cdot \prod_{\substack{y_{n+1} \in |\mathcal{A}_w|, \\ y_{n+1} \text{ is not a child of } y_n}} 1 \\ &= \prod_{\substack{y_{n+1} \in |\mathcal{A}_w|, \\ y_{n+1} \text{ is a child of } y_n}} \#w(\varphi^{(n+1)}[y_0/r]). \end{aligned}$$

Thus, we get a product only over the children of  $y_n$  and can drop the disjunct for  $i = n$  from the formula for the next step.

*Case 2:*  $n+1$  is even, so the  $(n+1)$ -st quantifier is an existential quantifier. Therefore, we get a  $\sum$ -operator from  $\#w(\varphi^{(n)}[y_0/r])$ , which is the same we get for an or-gate in the corresponding circuit. We now need to check over which values of  $y_{n+1}$  the sum runs:

The big conjunction can only be true if  $y_{n+1}$  is a child of  $y_n$ .

The big disjunction can also only be true if  $y_{n+1}$  is a child of  $y_n$ .

Thus, we directly get the sum

$$\sum_{\substack{y_{n+1} \in |\mathcal{A}_w|, \\ y_{n+1} \text{ is a child of } y_n}} \#w(\varphi^{(n+1)}[y_0/r]).$$

Here,  $\varphi^{(n+1)}$  does not drop a disjunct. This concludes the induction.

For  $\varphi^{(k)} = \text{trueLiteral}(y_k)$ , we get

$$\#w(\varphi^{(k)}) = \begin{cases} 1 & \text{if } \mathcal{A}_w \models_I \varphi^{(k)} \\ 0 & \text{else} \end{cases} = \begin{cases} 1 & \text{if } y_k \text{ is a true literal,} \\ 0 & \text{else.} \end{cases}$$

$\underline{\geq}$ : Let  $f$  be a function in  $\#\text{Win-FO}[\text{Arb}]$ . Let  $\tau$  be a vocabulary and  $\varphi$  be a formula over  $\tau \cup \tau_{\text{string}}$  together with the non-uniform family  $I = (I_n)_{n \in \mathbb{N}}$  of interpretations of the relation symbols in  $\tau$  a witness for  $f \in \#\text{Win-FO}[\text{Arb}]$ . Let  $k$  be the length of the quantifier prefix of  $\varphi$ . We now sketch how to construct a circuit family  $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$  that shows  $f \in \#\text{AC}^0$ . The gates of the circuit are  $\langle a_1, \dots, a_i \rangle$  with  $1 \leq i \leq k$  and  $a_j \in |\mathcal{A}_w|$  for all  $j$ . Each such gate has the meaning that we set the first  $i$  quantified variables to the values  $a_1, \dots, a_i$ . Therefore, for the  $i$ 'th quantifier of  $\varphi$  and for any choice of  $a_1, \dots, a_{i-1}$ ,  $\langle a_1, \dots, a_{i-1} \rangle$  is an and-gate if the quantifier was  $\forall$  and an or-gate if the quantifier was  $\exists$ . Also, if  $i \leq k$  we add as children to each such gate  $\langle a_1, \dots, a_{i-1}, a_i \rangle$  for all  $a_i \in |\mathcal{A}_w|$ .

On the lowest layer, where values have been assigned to all quantified variables, we add to every gate a circuit evaluating the quantifier-free part of  $\varphi$  for the choices made on that specific path. This can be done with a DNF which in each disjunct accesses the same variables. Thus, for any input there is exactly one proof tree for each DNF, if that DNF is true (and none otherwise). The size of these DNFs is constant, since they directly result from the quantifier-free part of  $\varphi$ . Handling the non-uniform family of interpretations  $I$  does not lead to problems, because for each fixed circuit  $C_n$  the input length and thus the specific interpretation  $I_n$  is fixed and thus only a Boolean function depending on the input bits has to be computed. The non-uniformity of the circuit family  $\mathcal{C}$  is used to build the different  $C_n$  depending on the different  $I_n$ .

Now by Lemma 13 it is clear that counting proof trees on this circuit family leads to the same function as counting winning strategies for the verifier for  $\mathcal{A}_w \models_I \varphi$ .  $\square$

## 4 The Uniform Case

Next we want to transfer this result to the uniform setting. In the direction from right to left we will have to show that the constructed circuit is uniform, which is straightforward. On the other hand, the following important point changes in the direction from left to right: We have to actually replace queries to  $C_{|w|}$  in the FO-sentence by the corresponding FO-formulae we get from the FO-interpretation which shows uniformness of  $\mathcal{C}$ . Since we introduce new quantifiers by this, we have to show how we can keep the counted value the same. That this is possible follows from the following lemma, proving that  $\#\text{Win-FO}[+, \times]$  is closed under FO-reductions (exact definitions follow).

**Lemma 15.** *Let  $\varphi$  be an  $\text{FO}[+, \times]$ -formula over some vocabulary  $\tau$ , and let  $I: \text{STRUC}[\sigma] \rightarrow \text{STRUC}[\tau]$  be an  $\text{FO}[+, \times]$ -interpretation. Then there is an  $\text{FO}[+, \times]$ -formula  $\varphi'$  over  $\sigma$  such that for all  $\mathcal{A} \in \text{STRUC}[\sigma]$ ,*

$$\#\text{Win}(\varphi', \mathcal{A}) = \#\text{Win}(\varphi, I(\mathcal{A})).$$

As already mentioned, this lemma yields an interesting closure property as a corollary, that is, closure under FO-reductions:

**Definition 16.** Let  $f, g : \{0, 1\}^* \rightarrow \mathbb{N}$ . We say that  $f$  is (many-one) first-order reducible to  $g$ , in symbols:  $f \leq^{\text{fo}} g$ , if there are vocabularies  $\sigma, \tau$  and an  $\text{FO}[+, \times]$  interpretation  $I : \text{STRUC}[\sigma] \rightarrow \text{STRUC}[\tau]$  such that for all  $\mathcal{A} \in \text{STRUC}[\sigma]$ :

$$f(\text{enc}_\sigma(\mathcal{A})) = g(\text{enc}_\tau(I(\mathcal{A}))).$$

**Corollary 17.** *On ordered structures with BIT, #Win-FO is closed under first-order reductions, that is, if  $f, g$  are functions such that  $g \in \# \text{Win-FO}$  and  $f \leq^{\text{fo}} g$ , then  $f \in \# \text{Win-FO}$ .*

Using Lemma 15 we can now establish the desired result in the FO-uniform setting.

**Theorem 18.**  $\text{FO}[+, \times]\text{-uniform } \# \text{AC}^0 = \# \text{Win-FO}[+, \times]$ .

*Proof (Sketch).*  $\subseteq$ : Let  $f \in \text{FO-uniform } \# \text{AC}^0$  via the circuit family  $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$  and the FO-interpretation  $I$  showing its uniformness. With the formula  $\varphi$  from the proof of  $\# \text{AC}^0 \subseteq \# \text{Win-FO}[\text{Arb}]$  this means we have for all  $w$ :

$$\begin{aligned} f(w) &= \text{number of proof trees of } C_{|w|} \text{ on input } w \\ &= \# \text{Win}(\varphi, C_{|w|}(w)), \end{aligned}$$

where  $C_{|w|}(w)$  is given as a  $\tau_{\text{circ}}$ -structure. From  $\varphi$  and  $I$  by Lemma 15 we get  $\varphi'$  over vocabulary  $\tau_{\text{string}}$  such that for all  $\mathcal{A}_w \in \text{STRUC}[\tau_{\text{string}}]$ :

$$\begin{aligned} \# \text{Win}(\varphi', \mathcal{A}_w) &= \# \text{Win}(\varphi, \underbrace{I(\mathcal{A}_w)}_{C_{|w|}(w)}) \\ &= f(w) \end{aligned}$$

$\supseteq$ : We can prove this analogously to  $\# \text{AC}^0 \supseteq \# \text{Win-FO}[\text{Arb}]$ . The only difference is that we need to show FO-uniformity of the circuit. Let  $f$  be a function in  $\# \text{Win-FO}$  with witness  $\varphi$ . We now need the formulae  $\varphi_{\text{universe}}, \varphi_{G_\wedge}, \varphi_{G_\vee}, \varphi_E, \varphi_I$  and  $\varphi_r$  defining the circuit. We do this by encoding gates in the circuit by suitable  $k$ -tuples over  $\{0, \dots, n\}$ , where  $k$  can be chosen to be the number of quantifiers in  $\varphi$ . The technical details will be given in the full paper.  $\square$

## 5 A Model-Theoretic Characterization of $\text{TC}^0$

We will now introduce the oracle class  $\text{AC}^0 \# \text{AC}^0$  as well as  $\text{FOCW}[\text{Arb}]$ , which is a variant of FO with counting. From the known connections between  $\text{TC}^0$  and  $\# \text{AC}^0$  and from the new connection between  $\# \text{AC}^0$  and  $\# \text{Win-FO}[\text{Arb}]$  we will then get a new model theoretic characterization for  $\text{TC}^0$ , the class of all languages accepted by Boolean circuits of polynomial size and constant depth with unbounded fan-in AND, OR, and MAJORITY gates, see [14]. First we want to define the above classes:

**Definition 19.**  $\text{AC}^{0\#\text{AC}^0}$  is the complexity class containing all languages decidable by  $\text{AC}^0$ -circuit families that may use gates computing bits of a fixed function from  $\#\text{AC}^0$ . More precisely, for each circuit family we fix a  $f \in \#\text{AC}^0$  and can use gates that are labeled with  $\#_i$ . Such a gate computes the Boolean function

$$\begin{aligned} f_i: \{0, 1\}^* &\rightarrow \{0, 1\} \\ \text{bin}(x) &\mapsto \text{BIT}(i, f(x)) \end{aligned}$$

The main result of this section will be a new characterization of the circuit class  $\text{TC}^0$  using a certain two-sorted logic.

**Definition 20.** Given a vocabulary  $\sigma$ , a  $\sigma$ -structure for  $\text{FOCW}[\text{Arb}]$  is a structure of the form

$$\langle \{a_0, \dots, a_{n-1}\}, \{0, \dots, n-1\}, (R_i)^A, +, \times, \underline{\min}, \underline{\max} \rangle,$$

where  $\langle \{a_0, \dots, a_{n-1}\}, (R_i)^A \rangle \in \text{STRUC}[\sigma]$ ,  $+$  and  $\times$  are the ternary relations corresponding to addition and multiplication in  $\mathbb{N}$  and  $\underline{\min}, \underline{\max}$  denote 0 and  $n-1$ , respectively. We assume that the two universes are disjoint. Formulas can have free variables of two sorts.

This logic extends the syntax of first order logic as follows:

- terms of the second sort:  $\underline{\min}, \underline{\max}$
- formulae:
  - (1) if  $t_1, t_2, t_3$  are terms of the second sort, then the following are (atomic) formulae:  $+(t_1, t_2, t_3), \times(t_1, t_2, t_3)$
  - (2) if  $\varphi(x, i)$  is a formula, then also  $\exists i\varphi(x, i)$  (binding the second-sort variable  $i$ )
  - (3) if  $Q$  is a quantifier prefix quantifying the first-sort variables  $\bar{x}$  and the second-sort variables  $\bar{i}$ ,  $\varphi(\bar{x}, \bar{i})$  is a quantifier-free formula and  $\bar{j}$  a tuple of second-sort variables, then the following is a formula:  $\#_{Q\varphi}(\bar{j})$

The semantics is clear except for  $\#_{Q\varphi}(\bar{j})$ . Let  $\mathcal{A}$  be an input structure and  $\bar{j}_0$  an assignment for  $\bar{j}$ . Then

$$\mathcal{A} \models \#_{Q\varphi}(\bar{j}_0) \iff_{\text{def}} \text{the } \text{val}(\bar{j}_0)\text{-th bit of } \#\text{Win}(Q\varphi, \mathcal{A}) \text{ is } 1$$

Here,  $\text{val}(\bar{j}_0)$  denotes the numeric value of the vector  $\bar{j}_0$  under an appropriate encoding of the natural numbers as tuples of elements from the second sort.

The types (1) and (2) of formulae in our definition are the same as in Definition 8.1 in [12, p. 142]. Additionally, our definition allows new formulae  $\#_{Q\varphi}(\bar{j})$ . These allow us to talk about the number of winning strategies for subformula  $\varphi$ . Note that these numbers can be exponentially large, hence polynomially long in binary representation; therefore we can only talk about them using some form of BIT predicate. Formulas of type (3) are exactly this: a BIT predicate applied to a number of winning strategies.

Our logic  $\text{FOCW}[\text{Arb}]$  thus gives FO with access to number of winning strategies, i.e., in  $\text{FOCW}[\text{Arb}]$  we can count in an exponential range. Libkin's logic

$\text{FO}(\text{Cnt})_{\text{All}}$  can count in the range of input positions, i.e., in a linear range. Nevertheless we will obtain the maybe somewhat surprising result that both logics are equally expressive on finite structures: both correspond to the circuit class  $\text{TC}^0$ .

**Theorem 21.** *On ordered structures,*

$$\text{TC}^0 = \text{FOCW}[\text{Arb}] = \text{AC}^{0\#\text{AC}^0}.$$

*Proof.* A central ingredient of the proof is the known equality  $\text{TC}^0 = \text{PAC}^0$  from [1]. Here,  $\text{PAC}^0$  is defined to be the class of languages  $L$  for which there exist functions  $f, h \in \#\text{AC}^0$  such that for all  $x, x \in L$  iff  $f(x) > h(x)$ .

The proof then consists of establishing the inclusions

$$\text{TC}^0 \subseteq \text{PAC}^0 \subseteq \text{FOCW}[\text{Arb}] \subseteq \text{AC}^{0\#\text{AC}^0} \subseteq \text{TC}^0,$$

and will be given in the full paper. □

*Remark 22.* In the full paper we will show that this result also holds in the uniform world. A central ingredient in the proof is that division, and thus iterated multiplication, can be done in uniform  $\text{TC}^0$  due to [9].

## 6 Conclusion

Arithmetic classes are of current focal interest in computational complexity, but no model-theoretic characterization for any of these was known so far. We addressed the maybe most basic arithmetic class  $\#\text{AC}^0$  and gave such a characterization, and, based on this, a new characterization of the (Boolean) class  $\text{TC}^0$ .

This immediately leads to a number of open problems:

- We mentioned the logical characterization of  $\#\text{P}$  in terms of counting assignments to free relations. We here count assignments to free function variables. Hence both characterizations are of a similar spirit. Can this be made more precise? Can our class  $\#\text{Win-FO}$  be placed somewhere in the hierarchy of classes from [13]?
- Can larger arithmetic classes be defined in similar ways? The next natural candidate might be  $\#\text{NC}^1$  which corresponds to counting paths in so called non-uniform finite automata [3]. Maybe this will lead to a descriptive complexity characterization.
- Still the most important open problem in the area of circuit complexity is the question if  $\text{TC}^0 = \text{NC}^1$ . While we cannot come up with a solution to this, it would be interesting to reformulate the question in purely logical terms, maybe making use of our (or some other) logical characterization of  $\text{TC}^0$ .

**Acknowledgements.** We are grateful to Lauri Hella (Tampere) and Juha Kontinen (Helsinki) for helpful discussion, leading in particular to Definition 20. We also thank the anonymous referees for helpful comments.

## References

1. Agrawal, M., Allender, E., Datta, S.: On  $TC^0$ ,  $AC^0$ , and arithmetic circuits. *J. Comput. Syst. Sci.* **60**(2), 395–421 (2000)
2. Barrington, D.A.M., Immerman, N., Straubing, H.: On uniformity within  $NC^1$ . *J. Comput. Syst. Sci.* **41**, 274–306 (1990)
3. Caussinus, H., McKenzie, P., Thérien, D., Vollmer, H.: Nondeterministic  $NC^1$  computation. *J. Comput. Syst. Sci.* **57**, 200–212 (1998)
4. Dawar, A.: The nature and power of fixed-point logic with counting. *ACM SIGLOG News* **2**(1), 8–21 (2015)
5. Ebbinghaus, H., Flum, J., Thomas, W.: *Mathematical Logic. Undergraduate Texts in Mathematics.* Springer, New York (1994)
6. Fagin, R.: Generalized first-order spectra and polynomial-time recognizable sets. In: Karp, R.M. (ed.) *Complexity of Computation*, vol. 7, pp. 43–73. SIAM-AMS Proceedings (1974)
7. Grädel, E.: Model-checking games for logics of imperfect information. *Theoret. Comput. Sci.* **493**, 2–14 (2013)
8. Gurevich, Y., Lewis, H.: A logic for constant-depth circuits. *Inf. Control* **61**, 65–74 (1984)
9. Hesse, W.: Division is in uniform  $TC^0$ . In: Orejas, F., Spirakis, P.G., Leeuwen, J. (eds.) *ICALP 2001. LNCS*, vol. 2076, pp. 104–114. Springer, Heidelberg (2001)
10. Immerman, N.: Languages that capture complexity classes. *SIAM J. Comput.* **16**, 760–778 (1987)
11. Immerman, N.: *Descriptive Complexity. Graduate Texts in Computer Science.* Springer, New York (1999)
12. Libkin, L.: *Elements of Finite Model Theory.* Springer, New York (2012)
13. Saluja, S., Subrahmanyam, K.V., Thakur, M.N.: Descriptive complexity of  $\#P$  functions. *J. Comput. Syst. Sci.* **50**(3), 493–505 (1995)
14. Vollmer, H.: *Introduction to Circuit Complexity - A Uniform Approach. Texts in Theoretical Computer Science. An EATCS Series.* Springer, New York (1999)