Descriptive Complexity of #AC^0 Functions*

Arnaud Durand[1], Anselm Haak[2], Juha Kontinen[3], and Heribert Vollmer[4]

1 **Université Paris Diderot, IMJ-PRG, CNRS UMR 7586, Case 7012, 75205 Paris cedex 13, France**
durand@logique.jussieu.fr
2 **Theoretische Informatik, Leibniz Universität Hannover, Germany**
haak@thi.uni-hannover.de
3 **Department of Mathematics and Statistics, University of Helsinki, Finland**
juha.kontinen@helsinki.fi
4 **Theoretische Informatik, Leibniz Universität Hannover, Germany**
vollmer@thi.uni-hannover.de

Abstract

We introduce a new framework for a descriptive complexity approach to arithmetic computations. We define a hierarchy of classes based on the idea of counting assignments to free function variables in first-order formulae. We completely determine the inclusion structure and show that #P and #AC^0 appear as classes of this hierarchy. In this way, we unconditionally place #AC^0 properly in a strict hierarchy of arithmetic classes within #P. We compare our classes with a hierarchy within #P defined in a model-theoretic way by Saluja et al. We argue that our approach is better suited to study arithmetic circuit classes such as #AC^0 which can be descriptively characterized as a class in our framework.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic

Keywords and phrases finite model theory, Fagin's theorem, arithmetic circuits, counting classes, Skolem function

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.20

1 Introduction

The complexity of arithmetic computations is a current focal topic in complexity theory. Most prominent is Valiant's class #P of all functions that count accepting paths of nondeterministic polynomial-time Turing machines. This class has interesting complete problems like counting the number of satisfying assignments of propositional formulae or counting the number of perfect matchings of bipartite graphs (the so-called permanent [15]).

The class #P has been characterized in a model-theoretic way by Saluja, Subrahmanyam and Thakur in [13]. Their characterization is a natural generalization of Fagin's Theorem: Given a first-order formula with a free relational variable, instead of asking if there exists an assignment to this variable that makes the formula true (NP = ESO), we now ask to count how many such assignments there are. In this way, the class #P is characterized: #P = #FO$^{\text{rel}}$. We use the superscript rel to denote that we are counting assignments to relational variables.

* Partially supported by DFG VO 630/8-1, grant 292767 of the Academy of Finland and grant ANR-14-CE25-0017-02 AGREG of the ANR.

From another point of view, the class #P can be seen as the class of those functions that can be computed by arithmetic circuits of polynomial size, i.e., circuits with plus and times gates instead of the usual Boolean gates (cf., e.g., [16]). This is why here we speak of *arithmetic computations*. In the following, all circuit complexity classes we are referring to will be FO-uniform classes, which means that there are FO-formulae describing the circuits for all input lengths (a formal definition will be given).

It is very natural to restrict the resource bounds of such arithmetic circuits. An important class defined in this way is the class #AC^0 of all functions computed by polynomial-size bounded-depth arithmetic circuits. It is interesting to note that #AC^0 and all analogous classes defined by arithmetic circuits, i.e., plus-times circuits, can also be defined making use of a suitable counting process: A witness that a Boolean circuit accepts its input is a so-called proof tree of the circuit, i.e., a minimal subtree of the circuit unwound into a tree, in which all gates evaluate to 1. Then the arithmetic class #AC^0, restricted to binary inputs, can be characterized as the counting class of all functions that count proof trees of (Boolean) AC^0 circuits. The correspondence between arithmetic computations and counting classes is explored in [3]. In this paper, we are mainly interested in these counting classes, and without further mention we use the notation #AC^0 in this vein.

There was no model-theoretic characterization of #AC^0, until it was recently shown in [10] that #AC^0 = #Π_1^{sk}, where #Π_1^{sk} means counting of possible Skolem functions for FO-formulae.

The aim of this paper is to compare the above two model-theoretic characterizations in order to get a unified view for both arithmetic circuit classes, #AC^0 and #P. This is done by noticing that the number of Skolem functions of an FO-formula can be counted as satisfying assignments to free function variables in a Π_1-formula. This gives rise to the idea to restate the result by Saluja et al counting functions instead of relations. We call our class where we count assignments to function variables #FO, in contrast to Saluja et al.'s #FO$^{\mathrm{rel}}$. In this setting, we get #P = #FO = #Π_1, which places both classes within #Π_1.

Furthermore, we show that #AC^0 actually corresponds to a syntactic fragment #Π_1^{prefix} of #Π_1 and, considering further syntactic subclasses of #FO defined by quantifier alternations, we get the inclusions

$$\#\Sigma_0 \subsetneq \#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{prefix}} \subsetneq \#\Pi_1 = \#\mathrm{FO} = \#\mathrm{P} \qquad (1)$$
$$\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1 = \#\mathrm{FO} = \#\mathrm{P}$$

Thus we establish (unconditionally, i.e., under no complexity theoretic assumptions) the complete structure of the alternation hierarchy within #FO and show where #AC^0 is located in this hierarchy.

Once we know that only universal quantifiers suffice to obtain the full class, i.e., #Π_1 = #P, it is a natural question to ask how many universal quantifiers are needed to express certain functions. We obtain the result that the hierarchy based on the number of universal variables is infinite; however, a possible connection to the depth hierarchy within #AC^0 remains open.

We see that counting assignments to free function variables instead of relation variables in first-order formulae leads us to a hierarchy of arithmetic classes suitable for a study of the power and complexity of the class #AC^0. The hierarchy introduced by Saluja et al. [13] does not seem suitable for such a goal.

This paper is organized as follows: In the next section, we introduce relevant concepts from finite model theory. Here, we also introduce the Saluja et al. hierarchy, and we explain the model-theoretic characterization of #AC^0. In Sect. 3 we introduce our new framework and the class #FO and its subclasses. In Sect. 4 we determine the full structure of the

alternation hierarchy within #FO and place $\#AC^0$ in this hierarchy, while in Sect. 5 we study the hierarchy defined by the number of universal variables in the $\#\Pi_1$-fragment. In Sect. 6 we turn to the hierarchy defined by Saluja et al. and show that the arithmetic class $\#AC^0$ is incomparable to all except the level-0 class and the full class of this hierarchy. Finally, we conclude in Sect. 7 with some open questions.

Our proofs make use of a number of different results and techniques, some stemming from computational complexity theory (such as separation of Boolen circuit classes or the time hierarchy theorem for nondeterministic RAMs), some from model theory (like closure of certain fragments of first-order logic under extensions or taking substructures) or descriptive complexity (correspondence between time-bounded NRAMs and fragments of existential second-order logic). Most techniques have to be adapted to work in our very low complexity setting (new counting reductions, use of the right set of built-in relations, etc.). Our paper sits right in the intersection of finite model theory and computational complexity theory.

2 Definitions and Preliminaries

In this paper we consider finite σ-structures where σ is a finite vocabulary consisting of relation and constant symbols. For a structure $\mathcal{A}$, $\mathrm{dom}(\mathcal{A})$ denotes its universe. We will always use structures with universe $\{0, 1, \ldots, n-1\}$ for some $n \in \mathbb{N} \setminus \{0\}$. Sometimes we will assume that our structures contain certain *built-in relations and constants*, e.g., $\leq^2$, SUCC^2, BIT^2 and min. In the following, we will always make it clear what built-in relations we allow. The interpretations of built-in symbols are fixed for any size of the universe as follows: $\leq^2$ is the $\leq$-relation on $\mathbb{N}$, min is 0, $\mathrm{SUCC}(i, j)$ is true, iff $i + 1 = j$, and $\mathrm{BIT}(i, j)$ is true, iff the i'th bit of the binary represention of j is 1. We will generally write $\mathrm{enc}_\sigma(\mathcal{A})$ for the binary encoding of a σ-structure $\mathcal{A}$. For this we assume the standard encoding (see e.g. [12]): Relations are encoded row by row by listing their truth values as 0's and 1's. Constants are encoded by the binary representation of their value and thus a string of length $\lceil \log_2(n) \rceil$. A whole structure is encoded by the concatenation of the encodings of its relations and constants except for the built-in numerical predicates and constants: These are not encoded, because they are fixed for any input length.

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

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

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 is the structure with universe $\{0, \ldots, n-1\}$, $\leq^2$ 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 is denoted by $\mathcal{A}_w$.

For any k, the fragments Σ_k and Π_k of FO are the classes of all formulae in prenex normal form with a quantifier prefix with k alternations starting with an existential or an universal quantifier, respectively.

In order to define uniformity of circuit families we need FO-interpretations, which are mappings between structures over different vocabularies.

▶ **Definition 1.** Let σ, τ be vocabularies, $\tau = (R_1^{a_1}, \ldots, R_r^{a_r})$. A first-order interpretation (or FO-interpretation)

$$I : \mathrm{STRUC}[\sigma] \to \mathrm{STRUC}[\tau]$$

is given by a tuple of FO-formulae $\varphi_0, \varphi_1, \ldots, \varphi_r$ over the vocabulary σ. For some k, φ_0 has k free variables and φ_i has $k \cdot a_i$ free variables for all $i \geq 1$. For each structure $\mathcal{A} \in \mathrm{STRUC}[\sigma]$, these formulae define the structure

$$I(\mathcal{A}) = \langle |I(\mathcal{A})|, R_1^{I(\mathcal{A})}, \ldots, R_r^{I(\mathcal{A})} \rangle \in \mathrm{STRUC}[\tau],$$

where the universe is defined by φ_0 and the relations by $\varphi_1, \ldots, \varphi_r$ in the following way:

$$|I(\mathcal{A})| = \{\langle b^1, \ldots, b^k \rangle \mid \mathcal{A} \vDash \varphi_0(b^1, \ldots, b^k)\}$$

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

We will now define the class #P and a model-theoretic framework in which the class can be characterized. Here, we follow [13] only changing the name slightly to emphasize that we are counting relations in this setting.

▶ **Definition 2.** A function $f\colon \{0,1\}^* \to \mathbb{N}$ is in #P, if there is a non-deterministic Turing-machine M such that for all inputs $x \in \{0,1\}^*$,

$$f(x) = \text{number of accepting computation paths of } M \text{ on input } x.$$

▶ **Definition 3.** A function $f\colon \{0,1\}^* \to \mathbb{N}$ is in $\#\mathrm{FO}^{\mathrm{rel}}$, if there is a vocabulary σ including built-in linear order $\leq$, and an FO-formula $\varphi(R_1, \ldots, R_k, x_1, \ldots, x_\ell)$ over σ with free relation variables $R_1, \ldots, R_k$ and free individual variables $x_1, \ldots, x_\ell$ such that for all $\mathcal{A} \in \mathrm{STRUC}[\sigma]$,

$$f(\mathrm{enc}_\sigma(\mathcal{A})) = |\{(S_1, \ldots, S_k, c_1, \ldots, c_\ell) \mid \mathcal{A} \vDash \varphi(S_1, \ldots, S_k, c_1, \ldots, c_\ell\}|.$$

If the input of f is not of this form, we assume f takes the value 0.

In the same fashion we define counting classes using fragments of FO, such as $\#\Sigma_{\mathrm{i}}^{\mathrm{rel}}$ and $\#\Pi_{\mathrm{i}}^{\mathrm{rel}}$ for arbitrary i. In [13] the following was shown for these classes (assuming order as the only built-in relation):

▶ **Theorem 4.** $\#\Sigma_0^{\mathrm{rel}} = \#\Pi_0^{\mathrm{rel}} \subset \#\Sigma_1^{\mathrm{rel}} \subset \#\Pi_1^{\mathrm{rel}} \subset \#\Sigma_2^{\mathrm{rel}} \subset \#\Pi_2^{\mathrm{rel}} = \#\mathrm{FO}^{\mathrm{rel}} = \#\mathrm{P}$.

Besides this theorem, it was also shown that the functions in $\#\Sigma_0^{\mathrm{rel}}$ can be computed in polynomial time.

To illustrate the definition just given, we repeat an example from Saluja et al. [13] that will also be important for us later.

▶ **Example 5.** We will show that #3DNF, the problem of counting the number of satisfying assignments of a propositional formula in disjunctive normal-form with at most 3 literals per disjunct, is in the class $\#\Sigma_1^{\mathrm{rel}}$. To do so, we use the vocabulary $\sigma_{\#3\mathrm{DNF}} = (D_0, D_1, D_2, D_3)$. Given a 3DNF-formula φ over variables V, we construct a corresponding σ-structure $\mathcal{A}_\varphi$ with universe V such that for any $x_1, x_2, x_3 \in V$, $D_i(x_1, x_2, x_3)$ holds iff $\bigwedge_{1 \leq j \leq i} \neg x_i \wedge \bigwedge_{i < j \leq 3} x_i$ appears as a disjunct. Now consider the following σ-formula with free relational variable T:

$$\Phi_{\#3\mathrm{DNF}}(T) = \exists x \exists y \exists z \quad \Big(\begin{array}{l} \big(D_0(x,y,z) \wedge T(x) \wedge T(y) \wedge T(z)\big) \\ \vee \big(D_1(x,y,z) \wedge \neg T(x) \wedge T(y) \wedge T(z)\big) \\ \vee \big(D_2(x,y,z) \wedge \neg T(x) \wedge \neg T(y) \wedge T(z)\big) \\ \vee \big(D_3(x,y,z) \wedge \neg T(x) \wedge \neg T(y) \wedge \neg T(z)\big) \Big) \end{array}$$

Observe that $\Phi_{\#3\mathrm{DNF}}$ is a Σ_1-formula. Evaluated on an input structure $\mathcal{A}_\varphi$, it expresses that an assignment to T defines a satisfying assignment of φ. Hence, the number of assignments $\mathbf{T}$ such that $\mathcal{A}_\varphi \models \Phi_{\#3\mathrm{DNF}}(\mathbf{T})$ is equal to the number of satisfying assignments of φ.

We will next recall the definition of Boolean circuits and counting classes defined using them. A circuit 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) bit of the input. Also, one gate is marked as the output gate. On any input, a circuit computes a Boolean function by evaluating all gates according to what they are marked with. The value of the output gate then is the function value for that input.

When we want circuits to work on different input lengths, we have to consider families of circuits: A family contains a 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:

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

The complexity classes in circuit complexity are classes of languages that can be decided by circuit families with certain restrictions to their depth and 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.
Above, we have not restricted the computability of the circuit $C_{|x|}$ from x in any way. This is called non-uniformity, which allows such circuit families to even compute non-recursive functions. Since we want to stay within #P, we need some notion of uniformity. For this, we first define the vocabulary for Boolean circuits as FO-structures:

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

where 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

We will now define FO-uniformity of Boolean circuits in general and the class AC^0.

▶ **Definition 6.** A circuit family $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ is said to be first-order uniform (FO-uniform) if there is an FO-interpretation

$$I : \text{STRUC}[\tau_{\text{string}} \cup (\text{BIT}^2)] \to \text{STRUC}[\tau_{\text{circ}}]$$

mapping any structure $\mathcal{A}_w$ over τ_{string} to the circuit $C_{|w|}$ given as a structure over the vocabulary τ_{circ}.

Note that by [4] this uniformity coincides with the maybe better known DLOGTIME-uniformity for many familiar circuit classes (and in particular for all classes studied in this paper).

▶ **Definition 7.** A language $L \subseteq \{0, 1\}^*$ is in AC^0, if there is an FO-uniform circuit family with constant depth and polynomial size accepting L.

It is known that the just given class coincides with the class FO of all languages definable in first-order logic [5, 12], i.e., informally: $\text{AC}^0 = \text{FO}$. This identity holds if our logical language includes the built-in relations of linear order and BIT. Though it is known that

linear order can be defined using BIT, we require that both are present in our language, because we consider very restricted quantifier prefixes where $\leq$ cannot be defined with BIT.

We will next define counting classes corresponding to Boolean circuit families. For a nondeterministic Turing machine, the witnesses we want to count are the accepting paths of the machine on a given input. Considering polynomial time computations, this concept gives rise to the class #P. A witness that a Boolean circuit accepts its input is a so-called *proof tree*, a minimal subtree of the circuit showing that it 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* (meaning that negations only occur directly in front of literals). A proof tree then is a subtree 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. This allows us to define the class #AC^0 as follows:

▶ **Definition 8.** A function $f\colon \{0,1\}^* \to \mathbb{N}$ is in #AC^0, if there is an FO-uniform circuit family $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ such that for all $w \in \{0,1\}^*$,

$$f(w) = \text{number of proof trees of } C_{|w|}(w).$$

As was shown in [10], there is a model-theoretic characterization of #AC^0. For this, let us define the Skolemization of an FO-formula φ in prenex normal form by removing all existential quantifiers and replacing each existentially quantified variable in the quantifier-free part of φ by a term consisting of a function application to those variables quantified universally to the left of the original existential quantifier. In other words, every existential variable is replaced by its so-called *Skolem function*. Now, #AC^0 contains exactly those functions that can be given as the number of Skolem functions for a given FO-formula.

▶ **Definition 9.** A function $f\colon \{0,1\}^* \to \mathbb{N}$ is in the class $\#\Pi_1^{\mathrm{sk}}$ if there is a vocabulary σ including built-in $\leq$, BIT and min and a first-order sentence φ over σ in prenex normal form

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

where ψ is quantifier-free such that for all $\mathcal{A} \in \mathrm{STRUC}[\sigma]$, $f(\mathrm{enc}_\sigma(\mathcal{A}))$ is equal to the number of tuples $(f_1, \ldots, f_k)$ of functions such that

$$\mathcal{A} \models \forall z_1 \ldots \forall z_{k-1} \; \psi(f_1, f_2(z_1), \ldots, f_k(z_1, \ldots, z_{k-1}), z_1, \ldots, z_{k-1}).$$

If the input of f is not of this form, we assume f takes the value 0.

This means that $\#\Pi_1^{\mathrm{sk}}$ contains those functions that, for a fixed FO-formula φ over some vocabulary σ, map an input w to the number of Skolem functions for φ on $\mathcal{A} = \mathrm{enc}_\sigma^{-1}(w)$.

▶ **Theorem 10.** $\#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{sk}}$.

The above mentioned result $\mathrm{FO} = \mathrm{AC}^0$ [5, 12] requires built-in order and BIT; hence it is no surprise that also for the theorem just given these relations are needed, and this is the reason why they also appear in Def. 9.

3 Connecting the Characterizations of #AC^0 and #P

We will now establish a unified view on the model-theoretic characterizations of both #AC^0 and #P. This will be done by viewing #AC^0 as a syntactic subclass of #FO. In Theorem 10 we characterized #AC^0 by a process of counting assignments to function variables in

FO-formulae, but only in a very restricted setting. It is natural to define the process of counting functions in a more general way, similar to the framework of [13], repeated here in Def. 3, where Saluja et al. count assignments to free relation variables in FO-formulae to obtain their characterization of #P.

▶ **Definition 11.** #FO is the class of all functions $f\colon \{0,1\}^* \to \mathbb{N}$ for which there is a vocabulary σ, including built-in $\leq$, BIT and min, and an FO-formula $\varphi(F_1,\dots,F_k,x_1,\dots,x_\ell)$ over σ with free function variables $F_1,\dots,F_k$ and free individual variables $x_1,\dots,x_\ell$ such that for all $\mathcal{A} \in \mathrm{STRUC}[\sigma]$,

$$f(\mathrm{enc}_\sigma(\mathcal{A})) = \left|\left\{(f_1,\dots,f_k,c_1,\dots,c_\ell) \mid \mathcal{A} \vDash \varphi(f_1,\dots,f_k,c_1,\dots,c_\ell)\right\}\right|.$$

If the input of f is not of this form, we assume f takes the value 0.

In the same fashion we define counting classes using fragments of FO, such as $\#\Sigma_\mathrm{i}$ and $\#\Pi_\mathrm{i}$ for arbitrary i. Note, that the free individual variables could also be seen as free function variables of arity 0.

We stress that our signatures in the above definition include symbols $\leq$, BIT, and min with their standard interpretations; as argued already several times, these built-ins are necessary in order to obtain a close correspondence between standard circuit classes like AC^0, TC^0 and first-order logic (but cf. results that consider weaker logics and relate them to presumably smaller non-standard complexity classes [6]). In contrast to our definition, Saluja et al. (see Def. 3) only use built-in order. Still, we will now see that both concepts, counting relations and counting function, are in fact equivalent as long as we use all of FO, even with different sets of built-in relations.

▶ **Theorem 12.** $\#\mathrm{FO}^{\mathrm{rel}} = \#\mathrm{FO} = \#\mathrm{P}$.

Proof. The inclusion $\#\mathrm{FO}^{\mathrm{rel}} \subseteq \#\mathrm{FO}$ is shown as follows: Let $f \in \#\mathrm{FO}^{\mathrm{rel}}$ via the formula φ containing free relation variables $R_1,\dots,R_k$. We can replace R_i by a function variable F_i of the same arity for all i. We then add a conjunct to the formula ensuring that for these functions only min and the element $x >$ min with $\forall y(y < x \to y = \mathrm{min})$ are allowed as function values. Then each occurrence of $R_i(\overline{z})$ can be replaced by $F_i(\overline{z}) = \mathrm{min}$.

The inclusion $\#\mathrm{FO} \subseteq \#\mathrm{P}$ is straightforward. The inclusion $\#\mathrm{P} \subseteq \#\mathrm{FO}^{\mathrm{rel}}$ was shown in [13]. ◀

Note that $\#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{sk}}$ does not directly arise from this definition by choosing an appropriate fragment of FO because of the restricted usage of the second-order variables in Def. 9. Still, we will characterize $\#\mathrm{AC}^0$ as a syntactic subclass of #FO as follows.

▶ **Definition 13.** Let $\#\Pi_1^{\mathrm{prefix}}$ be the class of all functions f for which there is a Π_1-formula $\varphi(\overline{G},\overline{x}) = \forall y_1 \dots \forall y_k \psi(\overline{G},\overline{x},y_1,\dots,y_k)$ over some vocabulary σ, where ψ is quantifier-free and in which all arity-a functions G (for any a) occur in ψ only as $G(y_1,\dots,y_a)$ such that for all $\mathcal{A} \in \mathrm{STRUC}[\sigma]$

$$f(\mathrm{enc}_\sigma(\mathcal{A})) = |\{(\overline{g},\overline{c}) \mid \mathcal{A} \vDash \varphi(\overline{g},\overline{c})\}|.$$

If the input of f is not of this form, we assume f takes the value 0.

▶ **Lemma 14.** $\#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{prefix}}$.

Proof. By Theorem 10 it suffices to show $\#\Pi_1^{\mathrm{sk}} = \#\Pi_1^{\mathrm{prefix}}$. We consider first the inclusion $\#\Pi_1^{\mathrm{sk}} \subseteq \#\Pi_1^{\mathrm{prefix}}$. Let $g \in \#\Pi_1^{\mathrm{sk}}$ via a formula φ as in Def. 9. Then we can simply replace

the occurrences of variables y_i in ψ by the corresponding function terms. The resulting formula is prefix-restricted as needed and directly shows $g \in \#\Pi_1^{\text{prefix}}$.

For $\#\Pi_1^{\text{prefix}} \subseteq \#\Pi_1^{\text{sk}}$, let $g \in \#\Pi_1^{\text{prefix}}$ via a formula φ. Since all function symbols occurring in φ are only applied to a unique prefix of the universally quantified variables, they can be seen as Skolem functions of suitable existentially quantified variables. Thus, we can replace the occurrences of the function symbols by new variables that are existentially quantified at adequate positions between the universally quantified variables. If for example, the input for a function was $x_1, \dots, x_\ell$, then the new variable is quantified after the part $\forall x_1 \dots \forall x_\ell$ of the quantifier prefix. Strict alternations in the quantifier-prefix, as required for $\#\Pi_1^{\text{sk}}$, can be achieved by adequately adding dummy-variables in between and forcing them to be equal to min. This yields a formula φ' that shows $g \in \#\Pi_1^{\text{sk}}$. ◀

4 An Alternation Hierarchy in #FO

In this section we study a hierarchy within #FO based on quantifier alternations. Interestingly, our approach allows us to locate #AC^0 in this hierarchy. First we note that the whole hierarchy collapses to a quite low class.

▶ **Theorem 15.** $\#\text{FO} = \#\Pi_1$.

Proof. Let $h \in \#\text{FO}$ via an FO-formula $\varphi(\overline{f}, \overline{x})$ in prenex normal form. We show how to transform φ to a Π_1-formula also defining h. As a first step, we change φ in such a way that for each existential variable instead of "there is an x" we say "there is a smallest x". Formally, this can be done with the following transformation:

$$\exists x \theta(x) \rightsquigarrow \exists x(\theta(x) \wedge \forall z(\neg\theta(z) \vee x < z \vee x = z))$$

applied recursively to all existential quantifiers in φ. Note that now for every satisfied $\exists$-quantifier there is exactly one witness.

For the sake of argument, suppose that after the above transformation and re-conversion to prenex normal form the formula $\varphi(\overline{f}, \overline{x})$ corresponds to

$$\varphi'(\overline{f}, \overline{x}) = \exists z_1 \forall y_1 \exists z_2 \dots \forall y_{\ell-1} \exists z_\ell \psi(\overline{f}, \overline{x}, z_1, \dots, z_\ell, y_1, \dots, y_{\ell-1})$$

where ψ is quantifier-free. Looking at the Skolemization of φ', our transformation ensures that every existentially quantified variable has a unique Skolem function. Thus,

$$\varphi''(\overline{f}, \overline{x}, g_1, \dots, g_\ell) = \forall y_1 \dots \forall y_{\ell-1} \psi(\overline{f}, \overline{x}, g_1, g_2(y_1), \dots, g_\ell(y_1, \dots, y_{\ell-1}), y_1, \dots, y_{\ell-1})$$

shows $h \in \#\Pi_1$. ◀

Next we look at the lowest class in our hierarchy and separate it from #AC^0.

▶ **Theorem 16.** $\#\Sigma_0 \subsetneq \#\text{AC}^0$.

Proof. We start by showing the inclusion. Certain observations in that proof will then almost directly yield the strictness. Let $f \in \#\Sigma_0$ via the quantifier-free FO-formula $\varphi(F_1, \dots, F_k, x_1, \dots, x_\ell)$ over some vocabulary σ, where $F_1, \dots, F_k$ are free function variables and $x_1, \dots, x_\ell$ are free individual variables, that is,

$$f(\text{enc}_\sigma(\mathcal{A})) = |\{(f_1, \dots, f_k, c_1, \dots, c_\ell) \mid \mathcal{A} \vDash \varphi(f_1, \dots, f_k, c_1, \dots, c_\ell)\}|.$$

Without loss of generality we can assume that in φ no nesting of functions occurs. If there is an occurrence of a function G as an argument for function H, then we can replace the

occurrence of G by a new free variable and force this variable to be equal to the function value. This ensures that there is only one unique assignment to this new free variable.

Let $A := \mathrm{dom}(\mathcal{A})$. For all i, let a_i be the arity of F_i and let m_i be the number of syntactically different terms that occur as inputs to F_i within φ. Let $e_{i1}, \dots, e_{im_i}$ be those terms in the order of their occurrence within φ and let $\varphi'(y_{11}, \dots, y_{1m_1}, \dots, y_{k1}, \dots, y_{km_k}, x_1, \dots, x_\ell)$ be φ after replacing for all i, j all occurrences of $F_i(e_{ij})$ by the new free variable y_{ij}. Let $m := \sum_i m_i$.

Considering a fixed assignment to the variables $x_1, \dots, x_\ell$, each e_{ij} has a fixed value. Thus, we can use free individual variables in order to count the number of assignments to all terms $F_i(e_{ij})$ for all (i, j). After that, all f_i have to be chosen in accordance with those choices to get the correct number of functions that satisfy the formula. Formally, this is done as follows:

$$\begin{aligned} f(\mathrm{enc}_\sigma(\mathcal{A})) &= \sum_{\overline{c} \in A^\ell} \sum_{\substack{(f_1,\dots,f_k)\in \\ A^{A^{a_1}} \times \cdots \times A^{A^{a_k}}}} [\mathcal{A} \vDash \varphi(f_1, \dots, f_k, c_1, \dots, c_\ell)] \\ &= \sum_{\overline{c} \in A^\ell} \sum_{\overline{d} \in A^m} \sum_{(f_1,\dots,f_k)\in G} [\mathcal{A} \vDash \varphi'(\overline{d}, \overline{c})], \end{aligned}$$

where $G := \{(f_1, \dots, f_k) \in A^{A^{a_1}} \times \cdots \times A^{A^{a_k}} \mid \forall (i, j) :\ \mathcal{A} \vDash\ d_{ij} = f_i(e_{ij})\}$.

Since $[\mathcal{A} \vDash \varphi'(\overline{d}, \overline{c})]$ does not depend on $(f_1, \dots, f_k)$, we can multiply by the cardinality of G instead of summing:

$$f(\mathrm{enc}_\sigma(\mathcal{A})) = \sum_{\substack{\overline{c} \in A^\ell, \\ \overline{d} \in A^m}} [\mathcal{A} \vDash \varphi'(\overline{d}, \overline{c})] \cdot |G|$$

Now we are in a position to show $f \in \#\mathrm{AC}^0$.

The sum only has polynomially many summands and thus is obviously possible in $\#\mathrm{AC}^0$.

For $[\mathcal{A} \vDash \varphi'(\overline{d}, \overline{c})]$, the circuit only has to evaluate a quantifier-free formula depending on an assignment that is given by the path from the root to the current gate. This is similar to the corresponding part of the proof of $\mathrm{FO} = \mathrm{AC}^0$ and thus can be done in $\mathrm{AC}^0 \subseteq \#\mathrm{AC}^0$.

For $|G|$ we first note that the total number of possible assignments for $\overline{f}$ is

$$|A^{A^{a_1}} \times \cdots \times A^{A^{a_k}}| = |A|^{\sum_i |A|^{a_i}}.$$

The definition of G fixes for each function f_i the function value on at most m_i inputs to be equal to some d_{ij}. This means, that the function value on at least $|A|^{a_i} - m_i$ inputs is not determined by the definition of G and can thus be freely chosen.

If for some (i, j), e_{ij} is semantically equal to $e_{ij'}$ for some $j' < j$, it has to hold that $d_{ij} = d_{ij'}$. Additionally, this reduces the amount of function values that are fixed by the d_{ij} by 1. To make this formal we define for any (i, j)

$$S_{ij} = \{j' \mid j' < j \text{ and } \mathcal{A} \vDash\ e_{ij} = e_{ij'}\}.$$

From the above considerations we get

$$|G| = [\bigwedge_{(i,j)} \bigwedge_{j'} (j' \in S_{ij}) \to d_{ij} = d_{ij'}] \cdot |A|^{\sum_i |A|^{a_i} - \sum_i m_i} \cdot |A|^{\sum_{ij} [S_{ij} \neq \emptyset]}.$$

Since the a_i and m_i are constants and S_{ij} is FO-definable, $|G|$ can be computed in $\#\mathrm{AC}^0$. This concludes the proof for $\#\Sigma_0 \subseteq \#\mathrm{AC}^0$.

Note that for any $\#\Sigma_0$-function f defined using a Σ_0-formula without free second-order variables, $f(w)$ is bounded polynomially in $|w|$ for all inputs w. On the other hand, the above proof shows that for any $\#\Sigma_0$-function f defined using a Σ_0-formula with at least one free second-order variable, there are constants $c_i > 0$ such that $f(w)$ divisible by $|w|^{\sum_i |w|^{c_i} - \text{const}}$ for all inputs w. Thus, the function $f(w) = |w|^{\lceil |w|/2 \rceil} \in \#\mathrm{AC}^0$ is not in $\#\Sigma_0$ which means $\#\Sigma_0 \neq \#\mathrm{AC}^0$. ◀

▶ **Theorem 17.** $\#\Pi_1^{\mathrm{sk}} \subsetneq \#\Pi_1$.

Proof. From the above we know that the left class is equal to $\#\mathrm{AC}^0$ and the right class is equal to #P. Strict inclusion now follows immediately from the following considerations: Let $\mathcal{F}$ be a class of functions $\{0,1\}^* \to \mathbb{N}$. Then the class $\mathsf{C} \cdot \mathcal{F}$ is the class of all languages L for which there are $f, g \in \mathcal{F}$ such that for all $x \in \{0,1\}^*$, $x \in L \Leftrightarrow f(x) > g(x)$. In [1] it was shown that $\mathrm{TC}^0 = \mathsf{C} \cdot \#\mathrm{AC}^0$. Also, it is well known that $\mathrm{PP} = \mathsf{C} \cdot \#\mathrm{P}$. Allender's separation $\mathrm{TC}^0 \neq \mathrm{PP}$ [2] now directly yields $\#\mathrm{AC}^0 \neq \#\mathrm{P}$. ◀

So far we have identified the following hierarchy:

$$\#\Sigma_0 \subsetneq \#\Pi_1^{\mathrm{sk}} = \#\mathrm{AC}^0 \subsetneq \#\Pi_1 = \#\mathrm{P}. \tag{2}$$

Next we turn to the class $\#\Sigma_1$ and show that it forms a different branch between $\#\Sigma_0$ and $\#\Pi_1$.

▶ **Lemma 18.** *There exists a function F which is in $\#\Pi_1^{\mathrm{sk}}$ but not in $\#\Sigma_1$.*

Proof. Let $\tau = \{E, c, d, \leq, \mathrm{BIT}, \min\}$ where E is a binary relation symbol and c, d are constant symbols. Let us consider the function F defined by the number of Skolem functions of variable z in the formula $\varphi = \forall x \forall y \exists z\ \psi(x, y, z)$ with

$$\psi = (E(x,y) \to z = c \vee z = d) \wedge (\neg E(x,y) \to z = c).$$

For a given τ-structure $\mathcal{A}$ with $c^{\mathcal{A}} \neq d^{\mathcal{A}}$, it is clear that:

$$F(\mathrm{enc}_\tau(\mathcal{A})) = |\{f \mid \mathcal{A} \models \forall x \forall y\ \psi(x, y, f(x,y))\}| = 2^{|E^{\mathcal{A}}|},$$

since each edge gives rise to two possible values for $z = f(x,y)$ and each non edge to only one value. Thus, $F \in \#\Pi_1^{\mathrm{sk}}$.

Suppose now that $F \in \#\Sigma_1$ i.e. that there exists $\phi(\overline{g}, \overline{x}) \in \Sigma_1$ such that for all τ-structures $\mathcal{G}$,

$$F(\mathrm{enc}_\tau(\mathcal{G})) = |\{(\overline{g}_0, \overline{a}) \mid \mathcal{G} \models \phi(\overline{g}_0, \overline{a})\}|$$

and in particular for $\mathcal{A}$ as above,

$$2^{|E^{\mathcal{A}}|} = F(\mathrm{enc}_\tau(\mathcal{A})) = |\{(\overline{g}_0, \overline{a}) \mid \mathcal{A} \models \phi(\overline{g}_0, \overline{a})\}|.$$

Now consider the following structure $\mathcal{A}'$ defined simply by extending $\mathrm{dom}(\mathcal{A}) = \{0, ..., n-1\}$ by two new elements, i.e., $\mathrm{dom}(\mathcal{A}') = \{0, ..., n+1\}$. Note that $E^{\mathcal{A}} = E^{\mathcal{A}'}$, hence the two structures have the same number of edges. To make the presentation simpler, suppose $\overline{g} = g$ and that the arity of g is one. Any given $g_0 : \mathrm{dom}(\mathcal{A}) \longrightarrow \mathrm{dom}(\mathcal{A})$, can be extended in several ways on the domain $\mathrm{dom}(\mathcal{A}')$ in particular as g_1 and g_2 below:

- $g_1(x) = g_0(x)$ for all $x \in \mathrm{dom}(\mathcal{A})$ and $g_1(n) = c$, $g_1(n+1) = d$.
- $g_2(x) = g_0(x)$ for all $x \in \mathrm{dom}(\mathcal{A})$ and $g_2(n) = d$, $g_2(n+1) = c$.

Formulas in Σ_1 are stable under extension of models so if $\overline{a}$ and g_0 are such that $\mathcal{A} \models \phi(g_0, \overline{a})$ then $\mathcal{A}' \models \phi(g_1, \overline{a})$ and $\mathcal{A}' \models \phi(g_2, \overline{a})$. Hence,

$$|\{(g', \overline{a}) \mid \mathcal{A}' \models \phi(g', \overline{a})\}| > |\{(g_0, \overline{a}) \mid \mathcal{A} \models \phi(g_0, \overline{a})\}|.$$

On the other hand, $F(\mathrm{enc}_\tau(\mathcal{A})) = F(\mathrm{enc}_\tau(\mathcal{A}'))$ holds, hence our assumpion that $\phi(g, \overline{x}) \in \Sigma_1$ defines F has led to a contradiction. ◀

For the opposite direction, we first show the following lemma.

▶ **Lemma 19.** *The function* #3DNF *is complete for* #P *under* AC^0*-Turing-reductions.*

Proof. A reduction of the #P-complete problem #3CNF to #3DNF is as follows: Given a 3CNF-formula φ over n variables, we first construct $\varphi' = \neg\varphi$. This is a 3DNF-formula. Obviously, the number of satisfying assignments of φ is equal to 2^n minus the number of satisfying assignments of φ'. Since this reduction can be computed by an AC^0-circuit and moreover #3CNF is #P-complete under AC^0-reductions (as follows from the standard proof of the NP-completeness of SAT), #3DNF is complete for #P under AC^0-Turing-reductions. ◀

▶ **Lemma 20.** *There exists a function F which is in $\#\Sigma_1$ but not in $\#\Pi_1^{\mathrm{sk}}$.*

Proof. Using FTC^0 to denote the functional version of TC^0, we first note that $\mathrm{FTC}^0 \neq \#\mathrm{P}$: For the sake of contradiction, assume $\mathrm{FTC}^0 = \#\mathrm{P}$. Making use of the complexity-theoretic operator C (see proof of Theorem 17), we obtain $\mathrm{PP} = \mathsf{C} \cdot \#\mathrm{P} \subseteq \mathsf{C} \cdot \mathrm{FTC}^0 = \mathrm{TC}^0$, but this is a contradiction to $\mathrm{TC}^0 \subsetneq \mathrm{PP}$ [2].

We now show this lemma by modifying the counting problem #3DNF to get a #P-complete function inside of $\#\Sigma_1$. If the reduction we use can be computed in FTC^0, the modified version of #3DNF can not be in $\#\Pi_1^{\mathrm{sk}} \subseteq \mathrm{FTC}^0$, because this would contradict $\mathrm{FTC}^0 \neq \#\mathrm{P}$.

Consider the vocabulary $\sigma_{3\mathrm{DNF}}$ and the formula $\Phi_{\#3\mathrm{DNF}}(T)$ from example 5. Let σ be the vocabulary extending $\sigma_{3\mathrm{DNF}}$ with built-in $\leq$, BIT and min. To get a function in $\#\Sigma_1$, we need to use a free function variable instead of the free relation variable T. Since we cannot use universal quantifiers, relations cannot be represented uniquely as functions of the same arity. In order to still get a #P-complete problem, we want to make sure that compared to #3DNF, the function value of our new counting function only differs from the one of #3DNF by a factor depending on the input length, not on the specific satisfying assignments. To achieve this, we encode the relation T as a function F as follows: interpret for all x an even function value $F(x)$ as $T(x)$ being false and an odd function value $F(x)$ as $T(x)$ being true. Thus, the number of 1's and 0's in a satisfying assignment do not influence the factor by which the new counting function differs from #3DNF.

Following this idea we define for all σ-structures $\mathcal{A}$

$$\#3\mathrm{DNF}^{\mathrm{func}}(\mathrm{enc}_\sigma(\mathcal{A})) = |\{f \mid \mathcal{A} \vDash \Phi_{\#3\mathrm{DNF}^{\mathrm{func}}}(f)\}|,$$

where $\Phi_{\#3\mathrm{DNF}^{\mathrm{func}}}(F)$ is $\Phi_{\#3\mathrm{DNF}}(T)$ after replacing for all variables x subformulae of the form $T(x)$ by $\mathrm{BIT}(\min, F(x))$. By definition, $\#3\mathrm{DNF}^{\mathrm{func}} \in \#\Sigma_1$.

We now want to reduce #3DNF to $\#3\mathrm{DNF}^{\mathrm{func}}$. Since the idea above only works if the universe has even cardinality, the first step of the reduction is doubling the size of the universe. Let $\mathcal{A}$ be a structure and $\mathcal{A}'$ the structure that arises from $\mathcal{A}$ by doubling the size of the universe. Let $A = \{0, \ldots, n-1\}$ and $A' = \{0, \ldots, 2n-1\}$ be their respective universes.

Each assignment for T with $\mathcal{A} \vDash \Phi_{\#3\mathrm{DNF}}(T)$ gives rise to the following set of assignments for f with $\mathcal{A}' \vDash \Phi_{\#3\mathrm{DNF}^{\mathrm{func}}}(f)$:

$$S_T = \{f \colon A' \to A' \mid \text{for all } x \in A : f(x) \equiv 1 \mod 2 \quad \Leftrightarrow \quad T(x)\}.$$

These sets are disjoint and by definition of $\Phi_{\#3\mathrm{DNF}^{\mathrm{func}}}(f)$ their union is equal to $\{f \mid \mathcal{A}' \vDash \Phi_{\#3\mathrm{DNF}^{\mathrm{func}}}(f)\}$. For each T, the functions f in S_T have n choices for $f(x)$, if $x \in A$ and $2n$ choices, if $x \notin A$. Thus, $|S_T| = |A|^{|A|} \cdot (2 \cdot |A|)^{|A|}$, yielding

$$\#3\mathrm{DNF}(\mathrm{enc}_{\sigma_{3\mathrm{DNF}}}(\mathcal{A})) = \frac{\#3\mathrm{DNF}^{\mathrm{func}}(\mathrm{enc}_{\sigma}(\mathcal{A}'))}{|A|^{2|A|} \cdot 2^{|A|}}.$$

Doubling the size of the universe can be done in FTC^0 by adding the adequate number of 0-entries in the encodings of all relations.

The term $|A|^{2|A|} \cdot 2^{|A|}$ can be computed in $\#\mathrm{AC}^0 \subseteq \mathrm{FTC}^0$ and division can be done in FTC^0 due to [11].

Since #3DNF is #P-complete under AC^0-Turing-reductions by Lemma 19, this means that #3DNF$^{\mathrm{func}}$ is #P-complete under TC^0-Turing-reductions. ◀

So Lemmas 18 and 20 show that $\#\Sigma_1$ and $\#\Pi_1^{\mathrm{sk}}$ are incomparable, and we obtain the inclusion chain $\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1 = \#\mathrm{P}$. Together with (2) we therefore obtain

$$\begin{array}{ccccc} & & \#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{prefix}} & & \\ \#\Sigma_0 & \subsetneq \atop \subsetneq & & \subsetneq \atop \subsetneq & \#\Pi_1 = \#\mathrm{FO} = \#\mathrm{P} \\ & & \#\Sigma_1 & & \end{array} \tag{1}$$

5 Hierarchy Based on the Number of Universal Variables

In this section we study another hierarchy in #FO based on syntactic restrictions, this time given by the number of universal variables.

Let Π_1^k denote the class of Π_1 formulae of the form

$$\forall x_1 \cdots \forall x_m \psi,$$

where $m \leq k$ and ψ is a quantifier-free formula. The function class corresponding to Π_1^k is denoted by $\#\Pi_1^k$. We will show that

$$\#\Pi_1^k \subsetneq \#\Pi_1^{k+1}, \tag{3}$$

for all $k \geq 1$. These results can be shown by applying a result of Grandjean and Olive which we will discuss next.

▶ **Definition 21.** We denote by $\mathrm{ESO}_f(k\forall)$ the class of ESO-sentences in Skolem normal form

$$\exists f_1 \ldots \exists f_n \forall x_1 \ldots \forall x_r \psi,$$

where $r \leq k$, and ψ is a quantifier-free formula.

It was shown in [9] that with respect to any finite signature σ

$$\mathrm{ESO}_f(k\forall) = \mathrm{NTIME}_{\mathrm{RAM}}(n^k),$$

where $\mathrm{NTIME}_{\mathrm{RAM}}(n^k)$ denotes the family of classes of σ-structures that can be recognized by a non-deterministic RAM in time $O(n^k)$. Note that by [8],

$$\mathrm{NTIME}_{\mathrm{RAM}}(n^k) \subsetneq \mathrm{NTIME}_{\mathrm{RAM}}(n^{k+1}).$$

These results can be used to show the strictness of the variables hierarchy (see (3)).

▶ **Theorem 22.** *Let $k \geq 1$. Then*

$$\#\Pi_1^k \subsetneq \#\Pi_1^{k+1}.$$

Proof. Let us fix $\sigma = \{<, \mathrm{BIT}, \min, P\}$, where P is unary. By the above there exists a sentence $\exists f_1 \cdots \exists f_n \psi \in \mathrm{ESO}_f(k+1\forall)[\sigma]$ defining a binary language L which cannot be defined by any sentence $\chi \in \mathrm{ESO}_f(k\forall)[\sigma]$. We claim that the function F associated with the formula $\psi(f_1, ..., f_n) \in \Pi_1^{k+1}$,

$$F(\mathrm{enc}_\sigma(\mathcal{A})) = |\{(f_1, ..., f_n) \mid \mathcal{A} \models \psi(f_1, ..., f_n)\}|,$$

is not a member of $\#\Pi_1^k$. For a contradiction, assume that $F \in \#\Pi_1^k$. Then there exists a formula $\chi(\overline{g}, \overline{y}) \in \Pi_1^k$ such that

$$F(\mathrm{enc}_\sigma(\mathcal{A})) = |\{(\overline{g}, \overline{y}) \mid \mathcal{A} \models \chi(\overline{g}, \overline{y})\}|$$

By the above, the sentence $\exists \overline{g} \exists \overline{y} \chi$ defines the language L, and hence contradicts the assumption that L cannot be defined by any $\mathrm{ESO}_f(k\forall)[\sigma]$-sentence. ◀

It is an interesting open question to study the relationship of $\#\mathrm{AC}^0$ with the classes $\#\Pi_1^k$.

6 #AC^0 compared to the classes from Saluja et al.

In this section we study the relationship of $\#\mathrm{AC}^0$ to the syntactic classes introduced in [13]. As in [13], these classes are defined assuming a built-in order relation only.

▶ **Theorem 23.**

- $\#\Sigma_0^{\mathrm{rel}} \subsetneq \#\mathrm{AC}^0$,
- *Let $\mathcal{C} \in \{\#\Sigma_1^{\mathrm{rel}}, \#\Pi_1^{\mathrm{rel}}, \#\Sigma_2^{\mathrm{rel}}\}$. Then the following holds: $\#\mathrm{AC}^0 \not\subseteq \mathcal{C}$ and $\mathcal{C} \not\subseteq \#\mathrm{AC}^0$.*

Proof. The proof of the inclusion $\#\Sigma_0^{\mathrm{rel}} \subsetneq \#\mathrm{AC}^0$ is analogous to the proof of Theorem 16 and is thus omitted.

For the second statement recall that $\#\Sigma_1^{\mathrm{rel}} \subset \#\Pi_1^{\mathrm{rel}} \subset \#\Sigma_2^{\mathrm{rel}}$. The claim $\mathcal{C} \not\subseteq \#\mathrm{AC}^0$ for $\mathcal{C} \in \{\#\Sigma_1^{\mathrm{rel}}, \#\Pi_1^{\mathrm{rel}}, \#\Sigma_2^{\mathrm{rel}}\}$ can be proven as follows: From Example 5 we know that $\#3\mathrm{DNF} \in \mathcal{C}$ and from Lemma 19 we know that #3DNF is #P-complete under AC^0-Turing-reductions. Now suppose $\#3\mathrm{DNF} \in \#\mathrm{AC}^0$. Then $\#\mathrm{P} \subseteq \mathrm{FAC}^{0\#\mathrm{AC}^0} \subseteq \mathrm{FTC}^0$ [10], contradicting $\mathrm{FTC}^0 \neq \#\mathrm{P}$, which was shown in the proof of Lemma 20. Hence $\#3\mathrm{DNF} \notin \#\mathrm{AC}^0$ and $\mathcal{C} \not\subseteq \#\mathrm{AC}^0$.

It remains to show $\#\mathrm{AC}^0 \not\subseteq \mathcal{C}$. We show this by an argument similar to the proof that #HAMILTONIAN is not in $\#\Sigma_2^{\mathrm{rel}}$, showing the separation of $\#\Sigma_2^{\mathrm{rel}}$ from #FO, see Theorem 2 in [13]. We will show that a very simple function f on encodings of τ_{string}-structures is not in $\mathcal{C}$. Define f as follows: $f(w) = 1$, if $|w|$ is even, and $f(w) = 0$ otherwise. Obviously $f \in \#\mathrm{AC}^0$. It now suffices to show that $f \notin \#\Sigma_2^{\mathrm{rel}}$. For contradiction, assume that $f \in \#\Sigma_2^{\mathrm{rel}}$ via a formula $\phi(\overline{R}, \overline{x}) \in \Sigma_2^{\mathrm{rel}}$, where

$$\phi(\overline{R}, \overline{x}) = \exists \overline{u} \forall \overline{v} \theta(\overline{R}, \overline{x}, \overline{u}, \overline{v}),$$

and θ is a quantifier-free formula. Let s and t be the lengths of the tuples $\overline{u}$ and $\overline{x}$, respectively. Let $n \geq s + t + 1$ be even and let $w \in \{0,1\}^n$. By the assumption, there exists $\overline{\mathbf{R}}$, $\overline{\mathbf{x}}$, $\overline{\mathbf{u}}$ such that

$$\mathcal{A}_w \models \forall \overline{v} \theta(\overline{\mathbf{R}}, \overline{\mathbf{x}}, \overline{\mathbf{u}}, \overline{v}).$$

By the choice of n, we can find $i \in \{0, .., n-1\}$ such that i does not appear in the tuples $\overline{\mathbf{u}}$ and $\overline{\mathbf{x}}$. Let $\mathcal{A}_{w'}$ denote the structure arising by removing the element i from the structure $\mathcal{A}_w$, and let $\overline{\mathbf{R}}^*$ denote the relations arising by removing tuples with the element i from $\overline{\mathbf{R}}$. By closure under substructures of universal first-order formulae, it follows that

$$\mathcal{A}_{w'} \models \forall \overline{v} \theta(\overline{\mathbf{R}}^*, \overline{\mathbf{x}}, \overline{\mathbf{u}}, \overline{v}),$$

implying that $f(\mathcal{A}_{w'}) \geq 1$. But $|w'|$ is odd and hence $f(\mathcal{A}_{w'}) = 0$ contradicting the assumption that the formula $\phi(\overline{R}, \overline{x})$ defined the function f. ◀

Last, we want to give another inclusion result between one of our classes and a class from the Saluja et al. hierarchy.

▶ **Lemma 24.** *There exists a function F which is in $\#\Sigma_1^{\text{rel}}$ but not in $\#\Sigma_1$.*

Proof. We prove that #3DNF is not in $\#\Sigma_1$ though it belongs to $\#\Sigma_1^{\text{rel}}$. As in Example 5, we use the vocabulary $\sigma_{\#3\text{DNF}} = (D_0, D_1, D_2, D_3)$ and consider the vocabulary σ extending $\sigma_{\#3\text{DNF}}$ with built-in linear order $\leq$, BIT and min. Suppose #3DNF is definable by a σ-formula $\Phi(\overline{g}, \overline{x}) \in \Sigma_1$. To a given DNF formula, φ, with $n \geq 2$ variables, one associates a σ-structure $\mathcal{A}_\varphi$ such that the number m of satisfying assignments of φ is equal to

$$m = |\{(\overline{g}_0, \overline{a}) \mid \mathcal{A}_\varphi \models \Phi(\overline{g}_0, \overline{a})\}|$$

Let $\{0, ..., n-1\}$ be the domain of $\mathcal{A}_\varphi$. Consider the structure $\mathcal{B}$ extending $\mathcal{A}_\varphi$ with one additional element n, correctly extending the numerical predicates. Structure $\mathcal{B}$ encodes a formula φ' whose number of satisfying assignments is obviously $2m$. Formulas in Σ_1 are stable by extension, so for any fixed $(\overline{g}_0, \overline{a})$ such that $\mathcal{A} \models \Phi(\overline{g}_0, \overline{a})$, any extension $\overline{g}'_0$ of $\overline{g}_0$ on the domain $\{0, ..., n\}$ of $\mathcal{B}$ is such that $\mathcal{B} \models \Phi(\overline{g}'_0, \overline{a})$. Each $g \in \overline{g}_0$ of arity $a \geq 1$ defined on $\{0, ..., n-1\}$ can be extended on $\{0, ..., n\}$ in at least $(n+1)^{\sum_{i=1}^{a} \binom{a}{i} n^{a-i}} \geq n+1$ ways. Hence:

$$|\{(\overline{g}'_0, \overline{a}) \mid \mathcal{B} \models \Phi(\overline{g}'_0, \overline{a})\}| \geq (n+1)m > 2m$$

contradicting the assumption that $\Phi(\overline{g}, \overline{x}) \in \Sigma_1$ defines #3DNF. ◀

7 Conclusion

In this paper we have started a descriptive complexity approach to arithmetic computations. We have introduced a new framework to define arithmetic functions by counting assignments to free function variables of first-order formulae. Compared to a similar definition of Saluja et al. where assignments to free relational variables are counted, we obtain a hierarchy with a completely different structure, different properties and different problems. The main interest in our hierarchy is that it allows the classification of arithmetic circuit classes such as #AC^0, in contrast to the one from Saluja et al.

We have only started the investigation of our framework, and many questions remain open for future research:

1. Sipser proved a depth hierarchy within the Boolean class AC^0 [14]. This hierarchy can be transferred into the context of arithmetic circuits: There is an infinite depth hierarchy within #AC^0. Does this circuit hierarchy lead to a logical hierarchy within $\#\Pi_1^{\text{sk}}$? Maybe it is possible to obtain a hierarchy defined by limiting the arity of the Skolem functions.

2. The connection between $\#\mathrm{AC}^0$ and the variable hierarchy studied in Sect.5 is not clear. We think it would be interesting to study if $\#\mathrm{AC}^0$ is fully contained in some finite level of this hierarchy.
3. One of the main goals of Saluja et al. in their paper [13] was to identify feasible subclasses of #P. They showed that $\#\Sigma_0^{\mathrm{rel}}$-functions can be computed in polynomial time, but even more interestingly, that functions from a certain higher class $\#R\Sigma_2$ allow a full polynomial-time randomized approximation scheme. Are there approximation algorithms or even schemes, maybe randomized, for some of the classes of our hierarchy?
4. The most prominent small arithmetic circuit class besides $\#\mathrm{AC}^0$ is problably the class $\#\mathrm{NC}^1$ [7]. Can it be characterized in our framework or by a natural extension of it, for example by allowing generalized quantifiers? The Boolean class NC^1 is obtained by first-order formulae with Lindström quantifiers for group word problems; i.e., we have, very informally, that $\mathrm{AC}^0 = \mathrm{FO}$ and $\mathrm{NC}^1 = \mathrm{FO} + \mathrm{GROUP}$, see [5, 16].
5. In Sect. 6, we clarified the inclusion relation between the class $\#\mathrm{AC}^0$ and all classes of the Saluja et al. hierarchy, and we gave a small number of examples for (non-)inclusion results between other classes from the two different settings. We consider it interesting to extend this systematically by studying the status of all further possible inclusions between classes from our hierarchy and classes of the Saluja et al. hierarchy.
6. We consider it interesting to study systematically the role of built-in relations. E.g., Saluja et al. define their classes using only linear order, and prove the hierarchy structure given in Theorem 4. It can be shown that by adding BIT, SUCC, min and max we obtain $\#\Pi_1^{\mathrm{rel}} = \#\mathrm{P}$. How does their hierarchy change when we generally introduce SUCC or BIT?

References

1 Manindra Agrawal, Eric Allender, and Samir Datta. On TC^0, AC^0, and arithmetic circuits. *Journal of Computer and System Sciences*, 60(2):395–421, 2000.

2 Eric Allender. The permanent requires large uniform threshold circuits. *Chicago J. Theor. Comput. Sci.*, 1999. URL: `http://cjtcs.cs.uchicago.edu/articles/1999/7/contents.html`.

3 Eric Allender. Arithmetic circuits and counting complexity classes. In *Complexity of Computations and Proofs, Quaderni di Matematica*, pages 33–72, 2004.

4 D. A. Mix Barrington and N. Immerman. Time, hardware, and uniformity. In *Proceedings 9th Structure in Complexity Theory*, pages 176–185. IEEE Computer Society Press, 1994.

5 D. A. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

6 C. Behle and K.-J. Lange. Fo[<]-uniformity. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 183–189. IEEE Computer Society Press, 2006. `doi:10.1109/CCC.2006.20`.

7 H. Caussinus, P. McKenzie, D. Thérien, and H. Vollmer. Nondeterministic NC^1 computation. *Journal of Computer and System Sciences*, 57:200–212, 1998.

8 Stephen A. Cook. A hierarchy for nondeterministic time complexity. In *Conference Record, Fourth Annual ACM Symposium on Theory of Computing*, pages 187–192. ACM, 1972.

9 Etienne Grandjean and Frédéric Olive. Graph properties checkable in linear time in the number of vertices. *J. Comput. Syst. Sci.*, 68(3):546–597, 2004.

10 Anselm Haak and Heribert Vollmer. A model-theoretic characterization of constant-depth arithmetic circuits. *CoRR*, abs/1603.09531, 2016. URL: `http://arxiv.org/abs/1603.09531`.

11 William Hesse. Division is in uniform TC^0. In *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, pages 104–114, 2001.

12 Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

13 Sanjeev Saluja, K. V. Subrahmanyam, and Madhukar N. Thakur. Descriptive complexity of #P functions. *Journal of Computer and System Sciences*, 50(3):493–505, 1995.

14 M. Sipser. Borel sets and circuit complexity. In *Proceedings 15th Symposium on Theory of Computing*, pages 61–69. ACM Press, 1983.

15 L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

16 Heribert Vollmer. *Introduction to Circuit Complexity – A Uniform Approach*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999.