Descriptive Complexity of #P Functions: A New Perspective

Arnaud Durand[a], Anselm Haak[b], Juha Kontinen[c], Heribert Vollmer[b]

[a]*Université Paris Diderot, IMJ-PRG, CNRS UMR 7586, Case 7012, 75205 Paris cedex 13, France*
durand@logique.jussieu.fr
[b]*Theoretische Informatik, Leibniz Universität Hannover, Appelstraße, D-30167, Germany*
(haak|vollmer)@thi.uni-hannover.de
[c]*Department of Mathematics and Statistics, University of Helsinki, Finland*
juha.kontinen@helsinki.fi

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. Furthermore, we show that some of our classes admit efficient approximation in the sense of FPRAS. We compare our classes with a hierarchy within #P defined in a model-theoretic way by Saluja et al and 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.

Keywords: finite model theory, arithmetic circuits, counting classes, Skolem function

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 (which is equivalent to computing the permanent of the adjacency matrix of such graphs [1]).

The class #P has been characterized in a model-theoretic way by Saluja, Subrahmanyam and Thakur in [2]. 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: $\#\mathrm{P} = \#\mathrm{FO}^{\mathrm{rel}}$. We use the superscript rel to denote that we are counting assignments to relational variables. The decision version PP of #P has been characterized from a descriptive complexity point of view in [3].

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., [4]). 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 $\#\mathrm{AC}^0$ of all functions computed by polynomial-size bounded-depth arithmetic circuits. It is interesting to note that $\#\mathrm{AC}^0$ and all analogous classes defined by arithmetic circuits, i.e., plus-times circuits, can also be defined making use of a suitable counting process for Boolean circuits in which negations only occur in the form of input gates being labeled as negated: A witness that such a Boolean circuit accepts its input is a so-called proof tree of the circuit, i.e., a subtree of the circuit unwound into a tree containing the output gate and for each contained gate a minimum number of its inputs that allow to deduce that it evaluates to 1. That also means that all contained input gates have to evaluate to 1. Then the arithmetic class $\#\mathrm{AC}^0$, 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 [5]. In this paper, we are mainly interested in these counting classes, and without further mention we use the notation $\#\mathrm{AC}^0$ in this vein.

There was no model-theoretic characterization of $\#\mathrm{AC}^0$, until it was recently shown in [6] that $\#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{Skolem}}$, where $\#\Pi_1^{\mathrm{Skolem}}$ means counting of possible Skolem functions for FO-formulae.

The aim of this paper is to compare the model-theoretic characterization obtained in [2] to that from [6] in order to get a unified view of both arithmetic circuit classes, $\#\mathrm{AC}^0$ 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 $\#\mathrm{FO}^{\mathrm{rel}}$. In this setting, we get $\#\mathrm{P} = \#\mathrm{FO} = \#\Pi_1$, which places both classes within $\#\Pi_1$.

Furthermore, we show that $\#\mathrm{AC}^0$ actually corresponds to a syntactic fragment $\#\Pi_1^{\mathrm{prefix}}$ of $\#\Pi_1$ and, considering further syntactic subclasses of #FO defined by quantifier alternations, we get the inclusions

$$\begin{tikzcd}
 & \#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{prefix}} \arrow[dr, phantom, "\subsetneq"] & \\
\#\Sigma_0 \arrow[ur, phantom, "\subsetneq"] \arrow[dr, phantom, "\subsetneq"] & & \#\Pi_1 = \#\mathrm{FO} = \#\mathrm{P}. \\
 & \#\Sigma_1 \arrow[ur, phantom, "\subsetneq"] &
\end{tikzcd} \qquad (1)$$

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

Once we know that only universal quantifiers suffice to obtain the full class, i.e., $\#\Pi_1 = \#\mathrm{P}$, 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 $\#\mathrm{AC}^0$ remains open.

We also study the question of which of the classes in our hierarchy are tractable. The main result we obtain is that all functions in Σ_1 posses a fully polynomial-time randomized approximation scheme (see e.g. [7]). For this, we consider a generalization of disjunctive normal-form we call "pseudo-DNF". A pseudo-DNF has identities of the form $f(\overline{a}) = b$ or $f(\overline{a}) \neq b$ as literals, where f is a function symbol, $\overline{a} \in \mathbb{N}^k$ for some $k \in \mathbb{N}$ and $b \in \mathbb{N}$. These are then evaluated over assignments to the function symbols. The main technical result we obtain in this area is an FPRAS for counting satisfying assignments for pseudo-k-DNF formulae. Then we show that every problem in $\#\Sigma_1$ can be reduced to such a problem. The concept of pseudo-DNF and the approximability may be of independent interest.

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 $\#\mathrm{AC}^0$. We show that the hierarchy introduced by Saluja et al. [2] is not suitable for such a goal, see Section 7.

This paper is an extended full version of an earlier conference paper[1]. In the

[1]Arnaud Durand, Anselm Haak, Juha Kontinen and Heribert Vollmer. Descriptive complexity

meantime, another paper appeared [8] that extended the framework by Saluja et al. In that paper, a so called quantitative logic is introduced, where arithmetic operators are introduced into second-order logic. This framework allowed for characterizations of the classes FP, #P and FPSPACE. Furthermore, in a recent paper [9], the study of descriptive complexity of arithmetic circuit classes was extended from $\#\mathrm{AC}^0$ to other small arithmetic circuit classes: Inspired by a work of Compton and Laflamme [10], variants of a certain recursion on predicates was introduced into FO. This yields characterizations of the Boolean classes NC^1, SAC^1 and AC^1 which extend to the respective counting classes. Another recent paper [11] studied functions counting assignments to free relational variables in Σ^1_1 formulae, including interesting connections to counting problems in team-based logics. In this way, a characterization of the class $\# \cdot \mathrm{NP}$ was obtained.

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 $\#\mathrm{AC}^0$. 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 $\#\mathrm{AC}^0$ in this hierarchy. In Sect. 5 we study the class $\#\Sigma_1$ with respect to efficient approximability and show that every function in this class admits an FPRAS. Finally, we conclude in Sect. 8 with some open questions. In Sect. 6 we study the hierarchy defined by the number of universal variables in the $\#\Pi_1$-fragment. In Sect. 7 we turn to the hierarchy defined by Saluja et al. and show that the arithmetic class $\#\mathrm{AC}^0$ is incomparable to all except the level-0 class and the full class of this hierarchy.

Our proofs make use of a number of different results and techniques, some stemming from computational complexity theory (such as separation of Boolean 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.

---

of $\#\mathrm{AC}^0$ functions. In *CSL*, volume 62 of *LIPIcs*, pages 20:1-20:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

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 such vocabularies σ, we denote by STRUC$[\sigma]$ the class of finite σ-structures. For a structure $\mathscr{A}$, $\mathrm{dom}(\mathscr{A})$ 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 representation of j is 1. We will generally write $\mathrm{enc}_\sigma(\mathscr{A})$ for the binary encoding of a σ-structure $\mathscr{A}$. 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 $\lfloor \log_2(n) \rfloor + 1$. 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 $\mathscr{A}_w$.

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.

We will now define the class #P and a model-theoretic framework in which the class can be characterized. Here, we follow [2] only changing the name slightly to emphasize that we are counting relations in this setting.

Definition 1. A function $f\colon \{0,1\}^* \to \mathbb{N}$ is in #P, if there is a non-deterministic polynomial-time 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 2. 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 $\varphi(R_1,\dots,R_k,x_1,\dots,x_\ell)$ over σ with free relation variables $R_1,\dots,R_k$ and free individual variables $x_1,\dots,x_\ell$ such that for all $\mathscr{A} \in \mathrm{STRUC}[\sigma]$,

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

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 [2] the following was shown for these classes (assuming order as the only built-in relation):

Theorem 3. $\#\Sigma_0^{\mathrm{rel}} = \#\Pi_0^{\mathrm{rel}} \subset \#\Sigma_1^{\mathrm{rel}} \subset \#\Pi_1^{\mathrm{rel}} \subset \#\Sigma_2^{\mathrm{rel}} \subset \#\Pi_2^{\mathrm{rel}} = \#\mathrm{FO}^{\mathrm{rel}} = \#\mathrm{P}$.

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. [2] that will also be important for us later.

Example 4. 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 $\mathscr{A}_\varphi$ 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 $\bigwedge_{1\leq j\leq i} \neg x_j \wedge \bigwedge_{i<j\leq 3} x_j$ appears as a disjunct. Now consider the following σ-formula with free relational variable T:

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

Observe that $\Phi_{\#3\mathrm{DNF}}$ is a Σ_1-formula. Evaluated on an input structure $\mathscr{A}_\varphi$, it expresses that an assignment to T defines a satisfying assignment of φ. Hence, the number of assignments $\mathbf{T}$ such that $\mathscr{A}_\varphi \models \Phi_{\#3\mathrm{DNF}}(\mathbf{T})$ 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. This means that negations can only occur as part of negated input gates and are not seperate gates. 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 $\mathscr{C} = (C_n)_{n\in\mathbb{N}}$ 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:

- $E(x,y)$: gate x is among the inputs of gate y
- $G_\wedge(x)$: gate x is an and-gate
- $G_\vee(x)$: gate x is an or-gate
- $B(x)$: gate x is an input gate of the circuit that evaluates to 1
- $r(x)$: x is the output gate of the circuit

Note that we are using a predicate for truth of input gates instead of specifying the associated input bit and whether it is negated or not. This is standard and ultimately both notions are equivalent when used to define uniform versions of the usual classes from circuit complexity.

We will now define FO-uniformity of Boolean circuits in general and the class AC^0. For a definition of FO-interpretations, also called FO-queries, see [12].

Definition 5. A circuit family $\mathscr{C} = (C_n)_{n\in\mathbb{N}}$ is said to be first-order uniform (FO-uniform) if there is an FO-interpretation

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

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

Note that by [13] 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 6. 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 AC^0 coincides with the class FO of all languages definable in first-order logic [14, 12], i.e., informally: $\mathrm{AC}^0 = \mathrm{FO}$. 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 $\#\mathrm{AC}^0$ as follows:

Definition 7. A function $f\colon \{0,1\}^* \to \mathbb{N}$ is in $\#\mathrm{AC}^0$, if there is an FO-uniform circuit family $\mathscr{C} = (C_n)_{n\in\mathbb{N}}$ such that for all $w \in \{0,1\}^*$,

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

As was shown in [6], there is a model-theoretic characterization of $\#\mathrm{AC}^0$. 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, $\#\mathrm{AC}^0$ contains exactly those functions that can be given as the number of Skolem functions for a given FO-formula.

Definition 8. A function $f\colon \{0,1\}^* \to \mathbb{N}$ is in the class $\#\Pi_1^{\text{Skolem}}$ 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

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

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

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

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

This means that $\#\Pi_1^{\text{Skolem}}$ 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 $\mathscr{A} = \mathrm{enc}_\sigma^{-1}(w)$.

The model-theoretic characterization of $\#\mathrm{AC}^0$ from [6] can now be stated as follows.

Theorem 9. $\#\mathrm{AC}^0 = \#\Pi_1^{\text{Skolem}}$.

The above mentioned result $\mathrm{FO} = \mathrm{AC}^0$ [14, 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. 8.

3. Connecting the Characterizations of $\#\mathrm{AC}^0$ and #P

We will now establish a unified view of the model-theoretic characterizations of both $\#\mathrm{AC}^0$ and #P. This will be done by viewing $\#\mathrm{AC}^0$ as a syntactic subclass of #FO. Theorem 9 characterizes $\#\mathrm{AC}^0$ 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 [2], repeated here in Def. 2, where Saluja et al. count assignments to free relation variables in FO-formulae to obtain their characterization of #P.

Definition 10. #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 $\mathscr{A} \in \mathrm{STRUC}[\sigma]$,

$$f(\mathrm{enc}_\sigma(\mathscr{A})) = \big|\big\{(f_1,\dots,f_k,c_1,\dots,c_\ell) \bigm| \mathscr{A} \vDash \varphi(f_1,\dots,f_k,c_1,\dots,c_\ell\big\}\big|.$$

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 [15]). In contrast to our definition, Saluja et al. (see Def. 2) only use built-in order. Still, we will now see that both concepts, counting relations and counting functions, are in fact equivalent as long as we use all of FO, even with different sets of built-in relations.

Theorem 11. $\#\mathrm{FO}^{\mathrm{rel}} = \#\mathrm{FO} = \#\mathrm{P}$.

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 $\forall y(y < x \to y = \min)$ are allowed as function values. Then each occurrence of $R_i(\overline{z})$ can be replaced by $F_i(\overline{z}) = \min$.

The inclusion $\#\mathrm{FO} \subseteq \#\mathrm{P}$ is straightforward. The inclusion $\#\mathrm{P} \subseteq \#\mathrm{FO}^{\mathrm{rel}}$ was shown in [2]. □

Note that $\#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{Skolem}}$ 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. 8. Still, we will characterize $\#\mathrm{AC}^0$ as a syntactic subclass of #FO as follows.

Definition 12. Let $\#\Pi_1^{\text{prefix}}$ be the class of all functions f for which there is a Π_1-formula $\varphi(\overline{G},\bar{x}) = \forall y_1 \ldots \forall y_k \psi(\overline{G},\bar{x},y_1,\ldots,y_k)$ over some vocabulary σ, where ψ is quantifier-free and in which all arity-a function symbols G (for any a) occur in ψ only as $G(y_1,\ldots,y_a)$ such that for all $\mathscr{A} \in \text{STRUC}[\sigma]$

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

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

Lemma 13. $\#\text{AC}^0 = \#\Pi_1^{\text{prefix}}$

Proof. By Theorem 9 it suffices to show $\#\Pi_1^{\text{Skolem}} = \#\Pi_1^{\text{prefix}}$. We consider first the inclusion $\#\Pi_1^{\text{Skolem}} \subseteq \#\Pi_1^{\text{prefix}}$. Let $g \in \#\Pi_1^{\text{Skolem}}$ via a formula φ as in Def. 8. 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 $\#\Pi_1^{\text{prefix}} \subseteq \#\Pi_1^{\text{Skolem}}$, 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 $x_1,\ldots,x_\ell$, then the new variable is quantified after the part $\forall x_1 \ldots \forall x_\ell$ of the quantifier prefix. Strict alternations in the quantifier-prefix, as required for $\#\Pi_1^{\text{Skolem}}$, 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 $g \in \#\Pi_1^{\text{Skolem}}$. □

4. An Alternation Hierarchy in #FO

In this section we study the quantifier alternation hierarchy of #FO. Interestingly, our approach allows us to locate $\#\text{AC}^0$ in this hierarchy. First we note that the whole hierarchy collapses to a quite low class.

Theorem 14. $\#\text{FO} = \#\Pi_1$

Proof. Let $h \in \#\text{FO}$ via an FO-formula $\varphi(\overline{F},\bar{x})$ 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:

$$\exists x \theta(x) \rightsquigarrow \exists x(\theta(x) \land \forall z(\neg\theta(z) \lor x \leq z))$$

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 conversion to prenex normal form with strict quantifier alternations the formula $\varphi(\overline{F},\overline{x})$ corresponds to

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

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,

$$\begin{aligned}\varphi''(\overline{F},\overline{x},G_1,\ldots,G_\ell) =& \forall y_1 \ldots \forall y_{\ell-1} \\ & \psi(\overline{F},\overline{x},G_1,G_2(y_1),\ldots,G_\ell(y_1,\ldots,y_{\ell-1}),y_1,\ldots,y_{\ell-1})\end{aligned}$$

shows $h \in \#\Pi_1$. $\square$

Next we look at the lowest class in our hierarchy and separate it from $\#\mathrm{AC}^0$.

Theorem 15. $\#\Sigma_0 \subsetneq \#\mathrm{AC}^0$

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 $\varphi(F_1,\ldots,F_k,x_1,\ldots,x_\ell)$ over some vocabulary σ, where $F_1,\ldots,F_k$ are free function variables and $x_1,\ldots,x_\ell$ are free individual variables, that is,

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

Without loss of generality we can assume that in φ no nesting of function symbols occurs by replacing each term $H(t_1,..,t_n)$ everywhere by $H(y_1,..,y_n)$, for some fresh variables $y_1,..,y_n$, and by adding a new conjunct $(\bigwedge_{1\leq i\leq n} y_i = t_i)$ to the formula. The new conjunct ensures that the interpretations of the new variables y_i are determined by the terms t_i and hence the total number of satisfying assignments remains equal to that of φ.

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

Considering a fixed assignment to the variables $x_1, \dots, x_\ell$, each $\overline{e}_{ij}$ has a fixed value. Thus, we can use free individual variables in order to count the number of assignments to all terms $F_i(\overline{e}_{ij})$ 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:

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

where $G_{\overline{c},\overline{d}} := \{(f_1,\dots,f_k) \in A^{A^{a_1}} \times \cdots \times A^{A^{a_k}} \mid \forall (i,j) : \mathscr{A} \vDash d_{ij} = f_i(\overline{e}_{ij})\}$ and $[\![P]\!]$ is the 0-1-value of proposition P.

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

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

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

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

For $[\![\mathscr{A} \vDash \varphi'(\overline{d},\overline{c})]\!]$, 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 $\mathrm{AC}^0 \subseteq \#\mathrm{AC}^0$ (viewing AC^0 as a class of characteristic functions of languages).

For $|G_{\overline{c},\overline{d}}|$ 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 $G_{\overline{c},\overline{d}}$ and can thus be freely chosen.

If for some (i, j), $\overline{e}_{ij}$ is semantically equal to $\overline{e}_{ij'}$ for some $j' < j$, it has to hold that $d_{ij} = d_{ij'}$. Additionally, if such a j' exists this reduces the number of function values that are fixed by the d_{ij} by 1. To make this formal we define for any $(\overline{c}, i, j)$

$$S_{\overline{c}ij} = \{j' \mid j' < j \text{ and } \mathscr{A} \vDash \overline{e}_{ij} = \overline{e}_{ij'}\}$$

where we use the extension of $=$ to tuples for simplicity. From the above considerations we get

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

Since the a_i and m_i are constants and S_{ij} is FO-definable, $|G_{\overline{c},\overline{d}}|$ 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 $\#\Sigma_0 \neq \#\mathrm{AC}^0$. $\square$

Theorem 16. $\#\Pi_1^{\text{Skolem}} \subsetneq \#\Pi_1$.

Proof. From the above we immediately get $\#\Pi_1^{\text{Skolem}} = \#\mathrm{AC}^0 \subsetneq \#\mathrm{P} = \#\Pi_1$, where the strict inclusion follows from the different closure properties of the two classes. For example, $\#\mathrm{AC}^0$ is not closed under a certain form of binomial coefficients as shown in [16] while #P is closed under this operation.

$\square$

So far we have identified the following hierarchy:

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

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

Lemma 17. *There exists a function f which is in $\#\Pi_1^{\text{Skolem}}$ 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 $\forall x \forall y \exists z\ \psi(x,y,z)$ with

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

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

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

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

Suppose now that $f \in \#\Sigma_1$ i.e. that there exists $\varphi(\overline{H},\overline{x}) \in \Sigma_1$ such that for all τ-structures $\mathscr{G}$,

$$f(\mathrm{enc}_\tau(\mathscr{G})) = |\{(\overline{h},\overline{a}) \mid \mathscr{G} \models \varphi(\overline{h},\overline{a})\}|$$

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

$$2^{|E^{\mathscr{A}}|} = f(\mathrm{enc}_\tau(\mathscr{A})) = |\{(\overline{h},\overline{a}) \mid \mathscr{A} \models \varphi(\overline{h},\overline{a})\}|.$$

Now consider the following structure $\mathscr{A}'$ defined simply by extending $\mathrm{dom}(\mathscr{A}) = \{0,\dots,n-1\}$ by a new element, i.e., $\mathrm{dom}(\mathscr{A}') = \{0,\dots,n\}$. Note that $E^{\mathscr{A}} = E^{\mathscr{A}'}$, hence the two structures have the same number of edges. To make the presentation simpler, suppose $\overline{H} = H$ and that the arity of H is one. Any given $h_0\colon \mathrm{dom}(\mathscr{A}) \longrightarrow \mathrm{dom}(\mathscr{A})$, can be extended in several ways on the domain $\mathrm{dom}(\mathscr{A}')$ in particular as h_1 and h_2 below:

- $h_1(x) = h(x)$ for all $x \in \mathrm{dom}(\mathscr{A})$ and $h_1(n) = c$.
- $h_2(x) = h(x)$ for all $x \in \mathrm{dom}(\mathscr{A})$ and $h_2(n) = d$.

Formulae in Σ_1 are preserved under extension of models so if $\overline{a}$ and h_0 are such that $\mathscr{A} \models \varphi(h_0,\overline{a})$ then $\mathscr{A}' \models \varphi(h_1,\overline{a})$ and $\mathscr{A}' \models \varphi(h_2,\overline{a})$. Hence,

$$|\{(h,\overline{a}) \mid \mathscr{A}' \models \varphi(h,\overline{a})\}| > |\{(h,\overline{a}) \mid \mathscr{A} \models \varphi(h,\overline{a})\}|.$$

On the other hand, $f(\mathrm{enc}_\tau(\mathscr{A})) = f(\mathrm{enc}_\tau(\mathscr{A}'))$ holds, hence our assumption that $\varphi(H,\overline{x}) \in \Sigma_1$ defines f has led to a contradiction. □

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

Lemma 18. *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 19. *There exists a function which is in* $\#\Sigma_1$ *but not in* $\#\Pi_1^{\text{Skolem}}$.

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 $\mathrm{FTC}^0 = \#\mathrm{P}$. For any class $\mathscr{F}$ of functions $\{0,1\}^* \to \mathbb{N}$ let $\mathsf{C} \cdot \mathscr{F}$ be the class of all languages L for which there are $f, g \in \mathscr{F}$ such that for all $x \in \{0,1\}^*$, $x \in L \Leftrightarrow f(x) > g(x)$. We obtain $\mathrm{PP} = \mathsf{C} \cdot \#\mathrm{P} \subseteq \mathsf{C} \cdot \mathrm{FTC}^0 = \mathrm{TC}^0$ which is a contradiction to $\mathrm{TC}^0 \subsetneq \mathrm{PP}$ [17]. Note that $\mathsf{C} \cdot \mathrm{FTC}^0 = \mathrm{TC}^0$ is obvious from the definition: Computing two FTC^0-functions and comparing them can be done in TC^0, while for the converse we can just use the characteristic function of any language in TC^0 and the constant-0 function.

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 $\#\Pi_1^{\mathrm{Skolem}} = \#\mathrm{AC}^0 \subseteq \mathrm{FTC}^0$, because this would contradict $\mathrm{FTC}^0 \neq \#\mathrm{P}$.

Consider the vocabulary σ_{3DNF} and the formula $\Phi_{\#\mathrm{3DNF}}(T)$ from example 4. Let σ be the vocabulary extending σ_{3DNF} 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 any relation T_0 interpreting T as a function f as follows: interpret for all x an even function value $f(x)$ as $T_0(x)$ being false and an odd function value $f(x)$ as $T_0(x)$ being true. If the size of the universe is even this ensures that the numbers 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 $\mathscr{A}$

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

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

We now want to reduce #3DNF to $\#\mathrm{3DNF}^{\mathrm{func}}$. 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 $\mathscr{A}$ be a structure and $\mathscr{A}'$ the structure that arises from $\mathscr{A}$ by doubling the size of the universe. Let $A = \{0, \dots, n-1\}$ and $A' = \{0, \dots, 2n-1\}$ be their respective universes. Each assignment T_0 for T with $\mathscr{A} \vDash \Phi_{\#\mathrm{3DNF}}(T_0)$ gives rise to the following set of assignments for f with $\mathscr{A}' \vDash \Phi_{\#\mathrm{3DNF}^{\mathrm{func}}}(f)$:

$$S_{T_0} = \{f \colon A' \to A' \mid \text{for all } x \in A : f(x) \equiv 1 \mod 2 \quad \Leftrightarrow \quad T_0(x)\}.$$

These sets are disjoint and by definition of $\Phi_{\#3\mathrm{DNF}^{\mathrm{func}}}(F)$ their union is equal to $\{f \mid \mathscr{A}' \vDash \Phi_{\#3\mathrm{DNF}^{\mathrm{func}}}(f)\}$. For each T_0, the functions f in S_{T_0} have n choices for $f(x)$, if $x \in A$ and $2n$ choices, if $x \notin A$. Thus, $|S_{T_0}| = |A|^{|A|} \cdot (2 \cdot |A|)^{|A|}$, yielding

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

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 [18].

Since #3DNF is #P-complete under AC^0-Turing-reductions by Lemma 18, this means that $\#3\mathrm{DNF}^{\mathrm{func}}$ is #P-complete under TC^0-Turing-reductions. □

So Lemmas 17 and 19 show that $\#\Sigma_1$ and $\#\Pi_1^{\mathrm{Skolem}}$ are incomparable, and we obtain the inclusion chain $\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1 = \#\mathrm{P}$. Together with (2) and Lemma 13 we therefore obtain

$$\begin{tikzcd} & \#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{prefix}} \arrow[dr, phantom, "\subsetneq"] & \\ \#\Sigma_0 \arrow[ur, phantom, "\subsetneq"] \arrow[dr, phantom, "\subsetneq"] & & \#\Pi_1 = \#\mathrm{FO} = \#\mathrm{P}. \\ & \#\Sigma_1 \arrow[ur, phantom, "\subsetneq"] & \end{tikzcd} \tag{1}$$

5. Feasibility of $\#\Sigma_1$

One of the main goals of Saluja et al. in their paper [2] 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 fully polynomial-time randomized approximation scheme. In this vein, we study the feasibility of the class $\#\Sigma_1$ in terms of approximability. We will show that every counting function in the class $\#\Sigma_1$ has a fully polynomial-time randomized approximation scheme (FPRAS). As an intermediate step we introduce the class of pseudo-DNF formulae as well as associated computational problems. It turns out that every problem in $\#\Sigma_1$ is reducible to the counting of satisfying assignments of a restricted pseudo-DNF formula and this problem—called #*k*PDNF—has an FPRAS. This approach is to a degree in analogy to the approach by Saluja et al showing that every problem in $\#\Sigma_1^{\mathrm{rel}}$ has an FPRAS. As a new tool we introduce the problem #*k*PDNF.

Definition 20. Let $f\colon \{0,1\}^* \to \mathbb{N}$ be a counting problem. f is said to have a *fully polynomial-time randomized approximation scheme* (*FPRAS*), if there is a randomized algorithm M working on inputs (x,ε) with $x \in \{0,1\}^*, \varepsilon \in \mathbb{Q}$ such that for all such inputs:

- $\mathscr{P}\Big[|M(x,\varepsilon) - f(x)| > \varepsilon \cdot |f(x)|\Big] < \frac{1}{4}$, where $M(x,\varepsilon)$ is the random variable describing the output of M on input (x,ε),
- the running time of M on input (x,ε) is bounded by a polynomial in $|x|$, $\frac{1}{\varepsilon}$.

We define the following type of reduction that preserves existence of FPRAS for languages:

Definition 21. Let f,g be counting problems. We say f is polynomial-time rational-product reducible to g if there are polynomial-time computable functions $r\colon \{0,1\}^* \to \{0,1\}^*$, $h\colon \{0,1\}^* \to \mathbb{Q}_+$, such that for all $w \in \{0,1\}^*$:

$$f(w) = h(w) \cdot g(r(w)).$$

Notation: $f \leq_{\mathbb{Q}\text{-pr}} g$.

The following Lemma is an obvious observation. We still give a proof for completeness.

Lemma 22. *Let f,g be counting problems, $f \leq_{\mathbb{Q}\text{-pr}} g$ and g has an FPRAS. Then f has an FPRAS.*

Proof. Let r and h be as in Definition 21 and

$$f(w) = h(w) \cdot g(r(w)).$$

The following is an FPRAS for f:

```
Require: $w$, $\varepsilon$
  Compute in polynomial time $r(w)$
  $t \leftarrow \varepsilon$-approximation of $g(r(w))$ using the FPRAS for $g$
  return $h(w) \cdot t$
```

The running-time of the FPRAS for g in this algorithm is polynomial in $|r(w)|$, $1/\varepsilon$ and hence also in $|w|$, $1/\varepsilon$. We now have

$$\begin{aligned}|g(r(w)) - t| \leq \varepsilon \cdot g(r(w)) &\Leftrightarrow h(w) \cdot |g(r(w)) - t| \leq \varepsilon \cdot h(w) \cdot g(r(w))\\ &\Leftrightarrow |h(w) \cdot g(r(w)) - h(w) \cdot t| \leq \varepsilon \cdot h(w) \cdot g(r(w))\\ &\Leftrightarrow |f(w) - h(w) \cdot t| \leq \varepsilon \cdot f(w).\end{aligned}$$

This means the error of the new FPRAS for f on input w is bounded by ε iff the error of the FPRAS for g on input $r(w)$ is bounded by ε. Furthermore, the new FPRAS does not use additional randomness beyond the randomness used by the FPRAS for g. This means that the error probability is still bounded by $\frac{1}{4}$, finishing the proof. $\square$

Let $\mathscr{F} = \{F_i \mid i \in \mathbb{N}\}$ be a countable set of function symbols and let $\mathrm{ar}(F_i)$ denote the arity of F_i. A pseudo-DNF formula is a formula φ of the following form:

$$\varphi = \bigvee_{i=1}^{m_1} \bigwedge_{j=1}^{m_2} \ell_{i,j},$$

where $\ell_{i,j} \triangleq F_{i,j}(\overline{a}_{i,j}) = b_{i,j}$ or $\ell_{i,j} \triangleq F_{i,j}(\overline{a}_{i,j}) \neq b_{i,j}$ with $F_{i,j} \in \mathscr{F}$, $\overline{a}_{i,j} \in \mathbb{N}^{\mathrm{ar}(f_i)}$ and $b_{i,j} \in \mathbb{N}$ for all $1 \leq i \leq m_1$ and $1 \leq j \leq m_2$.

We denote by $[n]_0$ the set of number from 0 to n. Let φ be a pseudo-DNF formula as above and $n \in \mathbb{N}$. An assignment of φ over $[n]_0$ is an assignment θ mapping each function symbol occurring in φ to an interpretation as a function over $[n]_0$ of corresponding arity. We say that θ is a satisfying assignment of φ if there is an i such that for all j we have

$$\theta(F_{i,j})(\overline{a}_{i,j}) = b_{i,j} \text{ if } \ell_{i,j} \text{ is the formula } F_{i,j}(\overline{a}_{i,j}) = b_{i,j}$$

and respectively for the case that $\ell_{i,j}$ is the formula $F_{i,j}(\overline{a}_{i,j}) \neq b_{i,j}$.

We now define the satisfiability problem for pseudo-DNF formulae as follows:

Problem:	PDNF-SAT
Input:	$(1^n, \varphi)$, where $n \in \mathbb{N}$, φ is a pseudo-DNF formula and all numbers occurring in φ are bounded by n
Question:	Is there a satisfying assignment of φ over $[n]_0$?

We also define the counting version of the above:

Problem:	#PDNF
Input:	$(1^n, \varphi)$, where $n \in \mathbb{N}$, φ is a pseudo-DNF formula and all numbers occurring in φ are bounded by n
Output:	number of satisfying assignments of φ over $[n]_0$

We will consider PDNFs with a bounded number of literals in each disjunct and bounded arity of function symbols. Let k-PDNF(m) be the class of PDNFs where each disjunct consists of at most k literals and the arity of each occurring function symbol is bounded by m. Furthermore, let k-PDNF(m)-SAT and #k-PDNF(m) be the above problems restricted to the respective inputs.

It can easily be seen that #k-PDNF(m) is #P-complete under subtractive reductions (see [19]): Membership is immediate. For hardness one can introduce the respective class of Pseudo-CNFs and the related problems. Encoding a Boolean assignment as a function of arity 1 it is straightforward to show that #k-CNF $\leq$ #k-PCNF(1) under parsimonious reductions. By using the same idea as for the reduction from #k-CNF to #k-DNF one can then show that #k-PDNF(1) is #P-hard under subtractive reductions.

Lemma 23. *For all $f \in \#\Sigma_1$ there are $k, m \in \mathbb{N}$ such that $f \leq_{\mathbb{Q}\text{-}pr}$* #$k$-PDNF$(m)$.

Proof. Let $f \in \#\Sigma_1$ via $\exists \overline{y} \psi(\overline{y}, \overline{z}, \overline{G})$, that is,

$$f(w) = |\{(\overline{g}, \overline{a}) \mid \mathscr{A}_w \vDash \exists \overline{y} \psi(\overline{y}, \overline{a}, \overline{g})\}|,$$

where by the introduction of existentially quantified new variables we may without loss of generality assume that ψ is quantifier-free, in DNF and all occurrences of function symbols in ψ are of the form $F(\overline{x}_1) = x_2$ or $F(\overline{x}_1) \neq x_2$, where F is a function symbol, $\overline{x}_1$ a tuple of first order variables of appropriate arity and x_2 a first order variable. Let t be a bound on the number of literals per disjunct in ψ. Let m be the highest arity among function symbols in $\overline{G}$. Furthermore, let p be the arity of $\overline{y}$ and q be the arity of $\overline{z}$. Let $\mathrm{dom}(\mathscr{A}_w)^p =: \{y_1, \ldots, y_{|w|^p}\}$ and $\mathrm{dom}(\mathscr{A}_w)^q =: \{z_0, \ldots, z_{|w|^q - 1}\}$.

Now, for each z_i, write the meaning of the existential quantifier out as a disjunction of polynomial size:

$$\exists \overline{y} \psi(\overline{y}, z_i, \overline{G}) \rightsquigarrow \bigvee_{j=1}^{|w|^p} \psi(y_j, z_i, \overline{G})$$

Define $\psi'_{z_i}(\overline{G})$ to be this formula after evaluating according to $\mathscr{A}_w$. That means, only a Boolean combination of atomic formulae of the form $G_i(\overline{b}) = c$ and $G_i(\overline{b}) \neq c$ remains (here $\overline{b}$ and c are constants), while all occurrences of symbols from the signature as well as equalities not involving free function variables have been evaluated.

We now use a tuple of new function symbols $\overline{H}$ to sum over the different values for z using the following disjoint disjunction:

$$\theta_{\mathscr{A}_w} := [\psi'_{z_0}(\overline{G}) \wedge \overline{H}(0) = \overline{0}] \vee \ldots \vee [\psi'_{z_{|w|^q-1}}(\overline{G}) \wedge \overline{H}(0) = z_{|w|^q-1}]$$

Let $\theta'_{\mathscr{A}_w}$ be $\theta_{\mathscr{A}_w}$ transformed to a PDNF in the obvious way, e.g. in the first disjunct $\overline{H}(0) = \overline{0}$ is added to all terms of $\psi'_{z_0}(\overline{G})$. The universe for the evaluation of $\theta'_{\mathscr{A}_w}$ is $\mathrm{dom}(\mathscr{A}_w) = [|w|-1]_0$. Since by using $\overline{H}$ we split for each different tuple assigned to $\overline{z}$, the number of satisfying assignments to the function symbols in $\theta'_{\mathscr{A}_w}$ is the sum of numbers of satisfying assignments for $\overline{G}, \overline{H}$ over all different values for $\overline{z}$. Since for $\overline{H}$ we only fix the value for input $\overline{0}$ (and the output for all other inputs can be chosen arbitrarily), we get

$$\#\mathrm{PDNF}(\theta'_{\mathscr{A}_w}, 1^{|w|-1}) = \left(|w|^{|w|^q-1}\right) \cdot f(w).$$

Note that we have bounds on the number of different function symbols, the arity of functions symbols and the number of literals in each term as follows: The tuple $\overline{H}$ is of size q and each function symbol in this tuple has arity 1. The arity of function symbols in $\overline{G}$ is bounded by m. Furthermore, the number of literals in each term in $\theta'_{\mathscr{A}_w}$ is bounded by $t+q$ and hence $\theta'_{\mathscr{A}_w}$ is a $(t+q)$-PDNF(m). Since $(\theta'_{\mathscr{A}_w}, 1^{|w|-1})$ and the function n^{n^q-1} (for constant q) are computable in polynomial time, this proves $f \leq_{\mathbb{Q}\text{-pr}} \#(t+1)$-PDNF$(m)$. □

The proof of Lemma 23 is inspired by Lemma 1 from [2]; however since they use a Boolean encoding of the different values of z, they need logarithmic clause width ($\#k \cdot \log \mathrm{DNF}$).

In order to show that $\#k$-PDNF(m) has FPRAS for all $k, m \in \mathbb{N}$, we first need the following Chernoff bound by Mitzenmacher and Upfal [20].

Lemma 24. *Let $X_1, \ldots, X_n$ be independent Poisson trials such that $Pr(X_i = 1) = p_i$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. For $0 < \delta < 1$,*

$$Pr(|X-\mu| \geq \delta\mu) \leq 2e^{-\mu\delta^2/3}.$$

Here, $\Pr(A)$ is the probability of event A and $\mathbb{E}[X]$ is the expected value of random variable X.

We will now prove that the number of satisfying assignments of k-PDNF(m)s can be approximated by an FPRAS for any $k, m \in \mathbb{N}$.

Lemma 25. *#k-*PDNF(m) *has* FPRAS *for all* $k,m \in \mathbb{N}$.

Proof. Let $k,m \in \mathbb{N}$. We give an FPRAS for #k-PDNF(m). Let $(\varphi, 1^n)$ be an input for #k-PDNF(m). This means that all numbers occurring in φ are bounded by n. Let ℓ be the number of function symbols occurring in φ and without loss of generality let the function symbols occurring in φ be F_i with arities $a_i = \mathrm{ar}(F_i)$ where $1 \leq i \leq \ell$. The total number of assignments of φ over $[n]_0$ is

$$\prod_{i=1}^{\ell} n^{n^{a_i}} = n^{\sum_{i=1}^{\ell} n^{a_i}}.$$

Now, if there is any satisfiable disjunct d, it fixes at most k function values. That means at least $n^{\sum_i n^{a_i} - k}$ assignments over $[n]_0$ satisfy d, which is at least a $\frac{1}{n^k}$ fraction of all assignments.

Let X be the $\{0,1\}$-valued random variable obtained by picking an assignment of φ uniformly at random and returning 1 iff that assignment satisfies φ. Let p be the probability of X being 1. Then $\mathbb{E}[X] = p$. We now apply Lemma 24: Let $t \in \mathbb{N}$ and $X_1, \ldots, X_t$ be independent instances of X. By Lemma 24 for every $1 > \varepsilon > 0$ it holds that

$$\Pr\left(\left|\sum_{i=1}^{t} X_i - \mathbb{E}[\sum_{i=1}^{t} X_i]\right| > \varepsilon \cdot \mathbb{E}[\sum_{i=1}^{t} X_i]\right) \leq 2e^{-\mathbb{E}[\sum_{i=1}^{t} X_i]\varepsilon^2/3}.$$

For $\varepsilon \geq 1$ we can simply use some fixed value < 1 instead to get the desired approximation. Dividing the inequality inside $\Pr(\cdot)$ by t and using $\frac{\mathbb{E}[\sum_{i=1}^{t} X_i]}{t} = \mathbb{E}[X] = p$, we obtain

$$\Pr\left(\left|\frac{\sum_{i=1}^{t} X_i}{t} - p\right| > \varepsilon \cdot p\right) \leq 2e^{-pt\varepsilon^2/3}.$$

This means that we can approximate p by $\frac{\sum_{i=1}^{t} X_i}{t}$ with the desired error probability by choosing t such that the right-hand side is bounded by $\frac{1}{4}$. This yields $t > \frac{3 \cdot \ln 8}{p\varepsilon^2}$. Since for $p > 0$ we have $p \geq \frac{1}{n^k}$, we choose

$$t = \frac{3\lceil \ln 8 \rceil \cdot n^k}{\varepsilon^2} + 1.$$

For $p = 0$ there will never be an error with any number of trials.

Now t is polynomially bounded in $\frac{1}{\varepsilon}$ and n and computable in polynomial time. We can now approximate the number of satisfying assignments by the total number of assignments times the above approximation of p. By these considerations, Algorithm 1 is an FPRAS for #k-PDNF(m). $\square$

Algorithm 1 FPRAS for k-PDNF(m)

```
Require: $(\varphi, 1^n)$, where $\varphi$ is a $k$-PDNF$(m)$, $\varepsilon$
  if $\varepsilon \geq 1$ then
      $\varepsilon = 0.9$
  end if
  sat $\leftarrow 0$
  $t \leftarrow \frac{3\lceil \ln 8 \rceil \cdot n^k}{\varepsilon^2} + 1$
  for $i = 1$ to $t$ do
      Pick an assignment $\theta$ of $\varphi$ over $[n]_0$ uniformly at random
      if $\theta$ satisfies $\varphi$ then sat $\leftarrow$ sat $+ 1$
      end if
  end for
  Let $\{F_1, \dots, F_\ell\}$ be the function symbols occurring in $\varphi$
  $a_i \leftarrow \mathrm{ar}(F_i)$
  return $n^{\sum_{i=1}^{\ell} n^{a_i}} \cdot \frac{\text{sat}}{t}$
```

6. 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

$$\forall x_1 \cdots \forall x_k \psi,$$

where 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 26. We denote by $\mathrm{ESO}_f(k\forall)$ the class of ESO-sentences in Skolem normal form

$$\exists F_1 \dots \exists F_n \forall x_1 \dots \forall x_k \psi,$$

where ψ is a quantifier-free formula.

It was shown in [21] 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 [22],

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

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

Theorem 27. *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 $\psi(F_1, ..., F_n) \in \Pi_1^{k+1}$,

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

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

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

By the above, the sentence $\exists \overline{G} \exists \overline{y} \chi$, where $\overline{y}$ is a tuple of arity-0 functions, defines the language L, and hence contradicts the assumption that L cannot be defined by any $\mathrm{ESO}_f(k\forall)[\sigma]$-sentence. $\square$

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

7. $\#\mathrm{AC}^0$ 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 [2]. As in [2], these classes are defined assuming a built-in order relation only.

Theorem 28.
- $\#\Sigma_0^{\mathrm{rel}} \subsetneq \#\mathrm{AC}^0$,

- *Let $\mathscr{C} \in \{\#\Sigma_1^{\text{rel}}, \#\Pi_1^{\text{rel}}, \#\Sigma_2^{\text{rel}}\}$. Then the following holds: $\#\text{AC}^0 \nsubseteq \mathscr{C}$ and $\mathscr{C} \nsubseteq \#\text{AC}^0$.*

Proof. The proof of the inclusion $\#\Sigma_0^{\text{rel}} \subsetneq \#\text{AC}^0$ is analogous to the proof of Theorem 15 and is thus omitted.

For the second statement recall from Theorem 3 that $\#\Sigma_1^{\text{rel}} \subset \#\Pi_1^{\text{rel}} \subset \#\Sigma_2^{\text{rel}}$. The claim $\mathscr{C} \nsubseteq \#\text{AC}^0$ for $\mathscr{C} \in \{\#\Sigma_1^{\text{rel}}, \#\Pi_1^{\text{rel}}, \#\Sigma_2^{\text{rel}}\}$ can be proven as follows: From Example 4 we know that $\#3\text{DNF} \in \mathscr{C}$ and from Lemma 18 we know that #3DNF is #P-complete under AC^0-Turing-reductions. Now suppose $\#3\text{DNF} \in \#\text{AC}^0$. Then $\#\text{P} \subseteq \text{FAC}^{0^{\#\text{AC}^0}} \subseteq \text{FTC}^0$ [6], contradicting $\text{FTC}^0 \neq \#\text{P}$, which was shown in the proof of Lemma 19. Hence $\#3\text{DNF} \notin \#\text{AC}^0$ and $\mathscr{C} \nsubseteq \#\text{AC}^0$.

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

$$\varphi(\overline{R}, \overline{x}) = \exists \overline{y} \forall \overline{z} \theta(\overline{R}, \overline{x}, \overline{y}, \overline{z}),$$

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

$$\mathscr{A}_w \models \forall \overline{z} \theta(\overline{\mathbf{R}}, \overline{a}, \overline{b}, \overline{z}).$$

By the choice of n, we can find $i \in \{0, .., n-1\}$ such that i does not appear in the tuples $\overline{a}$ and $\overline{b}$. Let $\mathscr{A}_{w'}$ denote the structure arising by removing the element i from the structure $\mathscr{A}_w$, 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

$$\mathscr{A}_{w'} \models \forall \overline{z} \theta(\overline{\mathbf{R}}^*, \overline{a}, \overline{b}, \overline{z}),$$

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

8. Conclusion

In this paper we have investigated 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 also show that all problems in the Σ_1-level of our hierarchy are efficiently approximable in the sense of FPRAS. To establish this result we also introduce so-called Pseudo-DNF-formulae which are formulae in disjunctive normal-form where literals are constraints on certain functions and show that the number of satisfying assignments to function symbols for such formulae is efficiently approximable with an FPRAS.

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 [23]. This hierarchy can be transferred into the context of arithmetic circuits: There is an infinite depth hierarchy within $\#AC^0$. This hierarchy is connected to the hierarchy obtained by restricting the number of different arities of Skolem functions in $\#\Pi_1^{\text{Skolem}}$. Similarly, Rossman proved a size hierarchy within AC^0 [24] over ordered graphs. Again, this hierarchy can be transferred into the context of arithmetic circuits and again there is likely a connection to a hierarchy within $\#\Pi_1^{\text{Skolem}}$, that is, the hierarchy obtained by restricting the number of variables in $\#\Pi_1^{\text{Skolem}}$. It could be interesting to further study these hierarchies within $\#\Pi_1^{\text{Skolem}}$ and make the connections to known hierarchies precise.

2. The connection between $\#AC^0$ and the variable hierarchy studied in Sect.6 is not clear. We think it would be interesting to study if $\#AC^0$ is fully contained in some finite level of this hierarchy.

3. In Sect. 7, we clarified the inclusion relation between the class $\#AC^0$ and all classes of the Saluja et al. hierarchy. 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. For example, it can be shown that $\#\Sigma_1^{\text{rel}} \not\subseteq \#\Sigma_1$ by showing that the function counting the number of satisfying assignments of a propositional formula in 3DNF is not in $\#\Sigma_1$.

4. 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 3. It can be shown that by adding BIT, SUCC, min and max we obtain $\#\Pi_1^{\text{rel}} = \#P$. How does their hierarchy change when we generally introduce SUCC or BIT?

5. As mentioned before, functions counting assignments to free relational

variables in Σ_1^1 formulae have been studied [11]. We believe a systematic study of counting free relational or functional variables in fragments of second-order logic might be worthwhile.

Acknowledgements

This work is partially supported by DFG VO 630/8-1, DAAD grant 57348395, grants 308099 and 308712 of the Academy of Finland and grant ANR-14-CE25-0017-02 AGREG of the ANR.

We thank the anonymous reviewers for their helpful comments.

Bibliography

[1] L. G. Valiant, The complexity of computing the permanent, Theor. Comput. Sci. 8 (1979) 189–201.

[2] S. Saluja, K. V. Subrahmanyam, M. N. Thakur, Descriptive complexity of #P functions, J. Comput. Syst. Sci. 50 (3) (1995) 493–505.

[3] J. Kontinen, A logical characterization of the counting hierarchy, ACM Trans. Comput. Log. 10 (1) (2009) 7:1–7:21.

[4] H. Vollmer, Introduction to Circuit Complexity - A Uniform Approach, Texts in Theoretical Computer Science. An EATCS Series, Springer, 1999.

[5] E. Allender, Arithmetic circuits and counting complexity classes, in: Complexity of Computations and Proofs, Quaderni di Matematica, 2004, pp. 33–72.

[6] A. Haak, H. Vollmer, A model-theoretic characterization of constant-depth arithmetic circuits, in: WoLLIC, Vol. 9803 of Lecture Notes in Computer Science, Springer, 2016, pp. 234–248.

[7] V. V. Vazirani, Approximation algorithms, Springer, 2001.

[8] M. Arenas, M. Muñoz, C. Riveros, Descriptive complexity for counting complexity classes, in: LICS, IEEE Computer Society, 2017, pp. 1–12.

[9] A. Durand, A. Haak, H. Vollmer, Model-theoretic characterization of boolean and arithmetic circuit classes of small depth, in: LICS, ACM, 2018, pp. 354–363.

[10] K. J. Compton, C. Laflamme, An algebra and a logic for NC^1, Inf. Comput. 87 (1/2) (1990) 240–262.

[11] A. Haak, J. Kontinen, F. Müller, H. Vollmer, F. Yang, Counting of teams in first-order team logics, CoRR abs/1902.00246 (2019).

[12] N. Immerman, Descriptive complexity, Graduate texts in computer science, Springer, 1999.

[13] D. A. Mix Barrington, N. Immerman, Time, hardware, and uniformity, in: Structure in Complexity Theory Conference, IEEE Computer Society, 1994, pp. 176–185.

[14] D. A. Mix Barrington, N. Immerman, H. Straubing, On uniformity within NC^1, J. Comput. Syst. Sci. 41 (3) (1990) 274–306.

[15] C. Behle, K. Lange, FO[<]-uniformity, in: IEEE Conference on Computational Complexity, IEEE Computer Society, 2006, pp. 183–189.

[16] M. Agrawal, E. Allender, S. Datta, On TC^0, AC^0, and arithmetic circuits, J. Comput. Syst. Sci. 60 (2) (2000) 395–421.

[17] E. Allender, The permanent requires large uniform threshold circuits, Chicago J. Theor. Comput. Sci. 1999 (1999).

[18] W. Hesse, Division is in uniform TC^0, in: ICALP, Vol. 2076 of Lecture Notes in Computer Science, Springer, 2001, pp. 104–114.

[19] A. Durand, M. Hermann, P. G. Kolaitis, Subtractive reductions and complete problems for counting complexity classes, Theor. Comput. Sci. 340 (3) (2005) 496–513.

[20] M. Mitzenmacher, E. Upfal, Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis, 2nd Edition, Cambridge University Press, 2017.

[21] E. Grandjean, F. Olive, Graph properties checkable in linear time in the number of vertices, J. Comput. Syst. Sci. 68 (3) (2004) 546–597.

[22] S. A. Cook, A hierarchy for nondeterministic time complexity, J. Comput. Syst. Sci. 7 (4) (1973) 343–353.

[23] M. Sipser, Borel sets and circuit complexity, in: STOC, ACM, 1983, pp. 61–69.

[24] B. Rossman, Average-case complexity of detecting cliques, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA (2010).