

Choiceless Polynomial Time, Counting and the Cai–Fürer–Immerman Graphs

Anuj Dawar

*University of Cambridge Computer Laboratory, William Gates Building,
J.J. Thomson Avenue, Cambridge, CB3 0FD, United Kingdom.*

David Richerby

*Department of Mathematics, University of Athens, Panepistimioupolis, GR157-84,
Athens, Greece.*

Benjamin Rossman

*Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge,
MA 02139, U.S.A.*

Abstract

We consider Choiceless Polynomial Time ($\tilde{\text{CPT}}$), a language introduced by Blass, Gurevich and Shelah, and show that it can express a query originally constructed by Cai, Fürer and Immerman to separate fixed-point logic with counting ($\text{IFP} + \text{C}$) from \mathbf{P} . This settles a question posed by Blass et al. The program we present uses sets of unbounded finite rank: we demonstrate that this is necessary by showing that the query cannot be computed by any program that has a constant bound on the rank of sets used, even in $\tilde{\text{CPT}}(\text{Card})$, an extension of $\tilde{\text{CPT}}$ with counting.

Key words: Finite model theory, descriptive complexity, choiceless polynomial time, counting quantifiers.

Email addresses: Anuj.Dawar@cl.cam.ac.uk (Anuj Dawar), davidr@math.uoa.gr (David Richerby), brossman@mit.edu (Benjamin Rossman).

¹ Funded by the European Social Fund and Greek National Resources — (EIEAEK II) PYTHAGORAS II. Research carried out at the University of Cambridge Computer Laboratory, supported by EPSRC grant GR/S06721.

² Research carried out during internships at Microsoft Research, One Microsoft Way, Redmond, WA 98052, U.S.A. and IBM Almaden Research Centre, 650 Harry Road, San Jose, CA 95120-6099, U.S.A.

1 Introduction

An important focus of the field of descriptive complexity is to provide logical characterizations of computational complexity with the aim of deploying logical and, in particular, model-theoretic methods to the study of complexity. The central open question in the area is whether there is a logic that exactly characterizes deterministic polynomial-time computability on relational structures. Such a logical characterization of **NP** is known through the work of Fagin [1], who showed that a class of structures is recognizable by a non-deterministic machine in polynomial time if, and only if, it is definable in existential second-order logic.

Immerman [2] and Vardi [3] independently showed that the polynomial-time computable properties of linearly ordered structures are definable in the fixed-point logic IFP (in this paper we will always use inflationary fixed-point logic, though we could equally well take least fixed-point logic, as it has equivalent expressive power [4,5]). IFP is too weak to express all polynomial-time properties of unordered structures. In particular, there is no formula of IFP that asserts that the size of a structure is even. Cai, Fürer and Immerman [6] showed that IFP + C, the extension of IFP with counting terms, is still too weak to express all polynomial-time properties, though it can easily express evenness and similar cardinality properties. The example constructed by Cai et al. gives, for each $k \in \mathbb{N}$, a pair of graphs, rich in automorphisms, that cannot be distinguished in $\mathcal{C}_{\infty\omega}^k$, the k -variable fragment of infinitary logic with counting quantifiers. It follows that the graphs are not distinguished by any formula of IFP + C, which is strictly contained within $\mathcal{C}_{\infty\omega}^\omega = \bigcup_{k < \omega} \mathcal{C}_{\infty\omega}^k$. There is, however, a single polynomial-time Turing machine that is able to distinguish the members of every pair. The logics we refer to are reviewed in Section 2 below and the reader may consult [7,8] for the proofs of the stated results and a more thorough account of the problem of providing a logical characterization of **P**.

Since the result of Cai et al., some logics have been proposed that extend the power of IFP + C while still defining only polynomial-time properties. A significant one is the logic with specified symmetric choice of Gire and Hoang [9] (see also [10,11] for an analysis of symmetric choice) which relies on a choice construct. It remains open whether this logic captures all polynomial-time computations. In an attempt to understand the role of choice in defining polynomial-time properties, Blass, Gurevich and Shelah [12] define *choiceless polynomial time*, $\tilde{\text{CPT}}$, a ‘logic’ based on a machine model, which they describe as an attempt to characterize what can be done without introducing choice. They show that this logic is strictly more expressive than IFP but is still unable to express cardinality properties such as the evenness of the number of elements in a structure. This leads to a natural extension of $\tilde{\text{CPT}}$ by means

of a counting mechanism. This extension, $\tilde{\text{CPT}}(\text{Card})$, is studied by the same authors, who show it to be strictly more expressive than $\text{IFP} + \text{C}$ [13]. The example that separates $\tilde{\text{CPT}}(\text{Card})$ from $\text{IFP} + \text{C}$ is a padded version of the problem defined by Cai et al. to prove that $\text{IFP} + \text{C}$ does not capture \mathbf{P} . Blass et al. leave open the question of whether there is any polynomial-time computable property that is not definable in $\tilde{\text{CPT}}(\text{Card})$ and, in particular, of whether the example of Cai et al. without padding is definable in the logic. In fact, the construction of Cai et al. yields not just a single example of a property that is not definable in $\text{IFP} + \text{C}$ but a general construction that gives, for each finite connected graph G with at least two vertices, a pair of graphs \mathfrak{G}^0 (referred to as ‘even’) and \mathfrak{G}^1 (‘odd’) which cannot be distinguished in \mathcal{C}^k if G has no separators of size k . (We refer to these graphs as *CFI graphs*.) Thus, using a suitable infinite sequence of graphs G the desired result is obtained. Indeed, we can also take G to be an ordered graph and this is the version of the construction used in [13]. For ordered graphs G , the graphs \mathfrak{G}^0 and \mathfrak{G}^1 are equipped with a linear pre-order.

In the present paper, we answer one question posed by Blass et al. by showing that, in the case when we begin with ordered graphs, there is an algorithm in $\tilde{\text{CPT}}$ that distinguishes \mathfrak{G}^0 from \mathfrak{G}^1 . The algorithm is based on the construction of objects (hereditarily finite sets) with a high degree of symmetry that are, nonetheless, able to determine the parity of the graph (i.e., distinguish between \mathfrak{G}^0 and \mathfrak{G}^1). The algorithm does not use counting but crucially relies on the use of hereditarily finite sets of unbounded rank. We show a corresponding lower bound by establishing that no $\tilde{\text{CPT}}(\text{Card})$ program using only objects of bounded rank can determine the parity of the CFI graphs, even when we begin with ordered graphs. The lower bound is based on an analysis of the automorphism structure of the graphs \mathfrak{G}^0 and \mathfrak{G}^1 , extending the techniques developed by Blass et al. in [12] to show that evenness is not definable in $\tilde{\text{CPT}}$.

The remainder of the paper is structured as follows. The next three sections cover the necessary background. In Section 2, we review the definitions of fixed-point logics, infinitary logics and logics with counting; Section 3 reviews the definition of the CFI graphs and Section 4 reviews the choiceless machine model and the definitions of $\tilde{\text{CPT}}$ and $\tilde{\text{CPT}}(\text{Card})$. Section 5 presents the proof that the CFI example can be solved by a $\tilde{\text{CPT}}$ program. The next three sections present the lower bound proof and are structured to follow the proof of Blass et al. that $\tilde{\text{CPT}}$ cannot express evenness on pure sets, but carried out on the CFI graphs, in the presence of counting. Section 6 shows that each $\tilde{\text{CPT}}(\text{Card})$ program on an input structure I can be translated into a formula of $\text{IFP} + \text{C}$ working on the extension of I with its hereditarily finite sets. In Section 7, we present a general result showing that any hereditarily finite set of fixed rank r activated in the course of any run of a $\tilde{\text{CPT}}(\text{Card})$ program on an input of size n is fixed, with respect to any Abelian automorphism group, by a set of elements (a ‘support’) of size $\mathcal{O}((\log n)^r)$. Section 8 shows that,

provided certain conditions (which, in particular, hold of the CFI graphs) are met, any distinctions that can be made in the structure with hereditarily finite sets in \mathcal{C}^k can be made in the original structure in \mathcal{C}^{mk} where m is the size of the support of the elements. It follows that no $\widetilde{\text{CPT}}(\text{Card})$ program that uses sets of rank bounded by some constant can determine the parity of CFI graphs. Concluding remarks appear in Section 9.

2 Preliminaries

We assume the standard definitions of a first-order vocabulary and a structure interpreting it. All vocabularies that we consider are finite and, unless otherwise stated, all structures are finite. An (m -ary) *query* is a map from structures over some fixed vocabulary σ to (m -ary) relations on the structures, that is closed under isomorphism. A 0-ary query is also called a *Boolean query*. Thus a first-order formula with up to m free variables defines an m -ary query and a sentence defines a Boolean query. We identify a Boolean query with the class of structures that it maps to true.

FO denotes first-order logic. In general a logic \mathcal{L} (such as FO or IFP) denotes both a set of formulae and the class of queries definable in that logic.

By closing first-order logic under an operator for forming the least fixed points of positive formulae, we obtain the logic LFP. This turns out to be equivalent in expressive power to the logic IFP which allows instead the formation of inflationary fixed points of arbitrary operators [4,5]. In the present paper, we use IFP in preference to LFP.

Formally, IFP is the logic obtained by adding to first-order logic the formula-formation rule that, if R is a new relation symbol, φ is a formula of IFP, \bar{x} a tuple of first-order variables and \bar{t} a tuple of terms, both of length equal to the arity of R , then $(\mathbf{ifp}_{R,\bar{x}}\varphi)(\bar{t})$ is a formula. The operator \mathbf{ifp} binds R and all occurrences of the variables in \bar{x} that appear in φ but any occurrences of these variables in \bar{t} remain free.

The semantics of IFP is given by the rule: $I \models (\mathbf{ifp}_{R,\bar{x}}\varphi)(\bar{t})$ if, and only if, the tuple \bar{t}^I is in the relation that is the limit of the sequence $\varphi^0 = \emptyset$; $\varphi^{i+1} = \varphi^i \cup \varphi(\varphi^i)$ where $\varphi(\varphi^i)$ is the relation defined in I by φ when R is interpreted as φ^i .

FO($\#$) is the extension of first-order logic with counting terms. To be precise, it has two sorts of variables: ordinary variables, which range over $|I|$, the domain of the structure I on which they are interpreted, and *number variables* which range over the numbers $\{0, \dots, \|I\|\}$ where $\|I\|$ is the cardinality of $|I|$. (We

also write $\|S\|$ for the cardinality of an unstructured set S , to avoid confusion with universes of structures.) The standard ordering and arithmetic operations on numbers are available in the language. In addition, for any formula φ we can form the term $\#x \varphi$ (where x is a variable of either sort), denoting the number of elements that satisfy $\varphi(x)$, modulo $\|I\| + 1$. (The modulo condition is required to allow the whole number domain to be counted.) IFP + C is the logic that extends FO($\#$) with the inflationary fixed-point operator. It was shown by Cai, Fürer and Immerman that IFP + C is strictly less expressive than polynomial time computation [6]. A detailed account of IFP + C and related results can be found in [14]; see also [8] for a textbook treatment.

Another way of adding counting to first-order logic is to introduce *counting quantifiers*. For each natural number i , we have a quantifier $\exists^{\geq i}$ where $I \models \exists^{\geq i} x \varphi$ if, and only if, there are at least i distinct elements $a \in I$ such that $I \models \varphi[a/x]$. While the extension of first-order logic with counting quantifiers is no more expressive than FO itself (in contrast to the situation with counting terms), the presence of these quantifiers does affect the number of variables that are necessary to express a query.

Let \mathcal{L}^k denote the fragment of FO consisting of those formulae which use only the variables x_1, \dots, x_k . \mathcal{C}^k denotes the k -variable fragment of first-order logic with counting quantifiers. $\mathcal{L}_{\infty\omega}^\infty$ is the extension of first-order logic with infinitary conjunctions and disjunctions. Let $\mathcal{L}_{\infty\omega}^k$ be the k -variable fragment of $\mathcal{L}_{\infty\omega}^\infty$ and let $\mathcal{C}_{\infty\omega}^\infty$ and $\mathcal{C}_{\infty\omega}^k$ be the corresponding logics with counting quantifiers. For each k , \mathcal{C}^k is more expressive than \mathcal{L}^k and $\mathcal{C}_{\infty\omega}^k$ is more expressive than $\mathcal{L}_{\infty\omega}^k$. Indeed, for each k , there are formulae of \mathcal{C}^1 that are not equivalent to any formula of $\mathcal{L}_{\infty\omega}^k$. Let $\mathcal{L}_{\infty\omega}^\omega = \bigcup_{k < \omega} \mathcal{L}_{\infty\omega}^k$ and $\mathcal{C}_{\infty\omega}^\omega = \bigcup_{k < \omega} \mathcal{C}_{\infty\omega}^k$. That is, $\mathcal{L}_{\infty\omega}^\omega$ consists of those infinitary formulae in which only finitely many distinct variables appear.

Infinitary logics are of interest because any formula of IFP is equivalent over finite structures to one of $\mathcal{L}_{\infty\omega}^\omega$. Similarly, IFP + C translates to $\mathcal{C}_{\infty\omega}^\omega$ (this involves translating counting terms into counting quantifiers) and most results establishing inexpressibility in IFP and IFP + C rely on this fact. Thus, the construction of Cai et al. gives a polynomial time query that is not definable in $\mathcal{C}_{\infty\omega}^\omega$.

Given two structures I and J and a logic \mathcal{L} , we write $I \equiv^{\mathcal{L}} J$ to indicate that no \mathcal{L} -formula distinguishes I and J . For finite I and J , $I \equiv^{\mathcal{L}^k} J$ if, and only if, $I \equiv^{\mathcal{L}_{\infty\omega}^k} J$ [15] and $I \equiv^{\mathcal{C}^k} J$ if, and only if, $I \equiv^{\mathcal{C}_{\infty\omega}^k} J$ [16]. Thus, to establish that a Boolean query is not definable in $\mathcal{C}_{\infty\omega}^\omega$ (and, *a fortiori*, not definable in IFP + C) it suffices to exhibit structures I_k and J_k for each k , that are distinguished by the query, with $I_k \equiv^{\mathcal{C}^k} J_k$. Hella has shown that $\equiv^{\mathcal{C}^k}$ is characterized by the *k-pebble bijection game* [17]. This is played by two players, the spoiler and the duplicator, on structures I and J , with k pairs of pebbles

$\{(x_i, y_i) : 1 \leq i \leq k\}$. For each move, the spoiler chooses a pair of pebbles (x_i, y_i) , the duplicator chooses a bijection $f : |I| \rightarrow |J|$ and the spoiler chooses $a \in |I|$ and places x_i on a and y_i on $f(a)$. If, after some move, the map $\bar{x} \mapsto \bar{y}$ is not a partial isomorphism, the spoiler wins; the duplicator wins infinite plays. The duplicator has a winning strategy if, and only if, $I \equiv^{C^k} J$.

3 The graphs

In this section, we describe the class of graphs that we will use in the rest of the paper. The graphs are originally due to Cai, Fürer and Immerman [6]; our presentation is essentially the same as that of Blass et al. [13] which, in turn, is an adaptation of Otto's presentation [14].

For a graph $G = (V, E)$, assumed to be finite, undirected and simple, $V(G) = V$, $\|G\| = \|V\|$ and $E(G) = E$. We write $E(v)$ for the set of edges incident on a vertex v .

Definition 1 *Let $G = (V, E)$ be a connected graph with at least two vertices. Let $\widehat{V} = \{v^X : v \in V \text{ and } X \subseteq E(v)\}$ and let $\widehat{E} = \{e^0, e^1 : e \in E\}$, where the v^X and e^i are new atoms (i.e., primitive entities considered not to be sets).*

$$G^* = \left(\widehat{V} \cup \widehat{E}, \left\{ \{v^X, e^1\} : e \in X \right\} \cup \left\{ \{v^X, e^0\} : e \in E(v) \setminus X \right\} \right).$$

For $v \in V$ of degree d , we write v^* for the associated set of 2^d vertices in \widehat{V} ; likewise, we write e^* for the set $\{e^0, e^1\}$.

Definition 2 *Let $G = (V, E, \leq)$ be an ordered connected graph with at least two vertices. \leq induces a lexicographic order on E , which we also write \leq . Let $\mathfrak{G} = (G^*, \preceq)$, where the linear pre-order \preceq is defined by*

- $v^X \preceq w^Y$ if, and only if, $v \leq w$;
- $v^X \preceq e^i$ for all v^X and e^i ;
- $e^i \preceq f^j$ if, and only if, $e \leq f$.

Any automorphism ρ of \mathfrak{G} must fix the set v^* for each $v \in V$ as it would otherwise not preserve \preceq . Also, it must fix the set e^* for each $e \in E$ as the vertices in different e^* 's are connected to vertices in different v^* 's. (Note that any individual v^* or e^* is not necessarily fixed pointwise.) It can be seen that the action of ρ is completely determined by the set $S \subseteq E$ given by

$$S = \left\{ e : \rho \text{ swaps } e^0 \text{ and } e^1 \right\}.$$

Indeed, for every $H = (V', E') \subseteq G$ (not necessarily an induced subgraph), we can define ρ_H to be the automorphism of \mathfrak{G} that flips exactly those edges in E' :

$$\rho_H : \begin{cases} e^i \mapsto e^{1-i} & \text{for } e \in E', i \in \{0, 1\} \\ e^i \mapsto e^i & \text{for } e \notin E' \\ v^X \mapsto v^{X \Delta (E' \cap E(v))}. & \end{cases}$$

Each ρ_H depends only on $E(H)$ and is an involution of \mathfrak{G} . $\text{Aut}(\mathfrak{G})$ is generated by the set of ρ_H where H contains a single edge.

The Cai–Fürer–Immerman graphs are subgraphs of \mathfrak{G} which have restricted automorphisms.

Definition 3 *Let $T \subseteq V$. For each $v \in T$, let $v^T = \{v^X : \|X\| \text{ is odd}\}$ and, for each $v \in V \setminus T$, let $v^T = \{v^X : \|X\| \text{ is even}\}$. \mathfrak{G}^T is the subgraph of \mathfrak{G} induced by $\widehat{E} \cup \bigcup_{v \in V} v^T$. \mathfrak{G}^T is even if $\|T\|$ is even and odd, otherwise.*

Since the ρ_H are automorphisms of \mathfrak{G} , the image of any $\mathfrak{G}^T \subseteq \mathfrak{G}$ under ρ_H must be an isomorphic copy of \mathfrak{G}^T and an induced subgraph of \mathfrak{G} . For a graph H , let $\text{odd}(H) \subseteq V(H)$ be the set of H 's vertices of odd degree. It is easy to see that $\|\text{odd}(H)\|$ must be even (otherwise, the sum of degrees is odd); call H an *even* subgraph of G if $\text{odd}(H) = \emptyset$. In fact, for every $H \subseteq G$ and $T \subseteq V$, $\rho_H(\mathfrak{G}^T) = \mathfrak{G}^{T \Delta \text{odd}(H)}$ and $\text{Aut}(\mathfrak{G}^T) = \{\rho_H : H \subseteq G \text{ is an even subgraph}\}$.

A proof of the following proposition can be found in, e.g., [13].

Proposition 4 *For connected G , $\mathfrak{G}^S \cong \mathfrak{G}^T$ if, and only if, $\|S\| \equiv \|T\| \pmod{2}$.*

As such, the even and odd versions of \mathfrak{G}^T are uniquely defined up to isomorphism: call these graphs \mathfrak{G}^0 and \mathfrak{G}^1 , respectively. The *parity* of a CFI graph \mathfrak{G}^T is the parity of $\|T\|$.

4 $\widetilde{\text{CPT}}(\text{Card})$

The $\widetilde{\text{CPT}}$ model of computation is introduced by Blass, Gurevich and Shelah in [12] and extended with a counting mechanism by the same authors in [13] to give $\widetilde{\text{CPT}}(\text{Card})$.

We summarize the computation model here but the reader should consult these two references, particularly [12], for a full description. Our presentation is substantially the same as that of Blass et al.: the only significant difference is that we represent numbers as a linearly-ordered set of atoms, which is the way they are represented in $\text{IFP} + \text{C}$, whereas Blass et al. code them as von

Neumann ordinals. This change is motivated by our interest in the rank of the sets used in a computation but has no other material effect on the computation model.

Conventional models of computation, such as Turing machines, are sequential devices operating on strings. Because the machines operate on strings, relational structures must be encoded to be used as inputs. This coding is straightforward but is problematic because it is possible that a machine might accept some codings of a structure and reject others. In contrast, $\tilde{\text{CPT}}$ programs are based on Gurevich's abstract state machines [18], so are parallel and operate directly on relational structures.

Given an input structure I of vocabulary σ , a $\tilde{\text{CPT}}$ program operates on hereditarily finite sets over $|I|$, with the elements of $|I|$ viewed as atoms (objects that are not sets). $\text{HF}(I)$ is the least set having as members all elements of $|I|$ and all its own finite subsets. We abuse notation and also write $\text{HF}(I)$ for the extension of I with universe $\text{HF}(I)$ and the additional relation \in and constant \emptyset with the obvious interpretations. Any automorphism η of I naturally induces an automorphism of $\text{HF}(I)$; we will usually write η for this induced automorphism, also.

An object $x \in \text{HF}(I)$ is *transitively-closed* if, whenever $y \in x$ and $z \in y$, we have $z \in x$. The *transitive closure* of x is $\text{TC}(x)$, the least transitively-closed set containing x . The *rank* of an object $x \in \text{HF}(I)$ is defined inductively: \emptyset and the elements of $|I|$ have rank 0; the rank of a non-empty set x is one greater than the maximal rank of its members.

A $\tilde{\text{CPT}}$ program proceeds by making parallel updates to a series of 'dynamic functions' via rules that are iterated until the distinguished nullary dynamic function Halt is set to true. At this point, the program is deemed to accept if, and only if, the distinguished nullary dynamic function Output is true.

The vocabulary of a program consists of two parts: the input vocabulary σ , which is assumed to be purely relational, and the vocabulary δ of dynamic function names, including the nullary functions Halt and Output .

4.1 States

A computation of a program with variables v_1, \dots, v_k over input structure I is a finite or countable sequence of *states* $\langle S_i : i < \kappa \rangle$ for some $\kappa \leq \omega$.

States of $\tilde{\text{CPT}}$ programs are expansions of $\text{HF}(I)$ adding all functions in δ and constants v_1, \dots, v_k , whose interpretations give the values of the dynamic functions and the variables, respectively. States of $\tilde{\text{CPT}}(\text{Card})$ programs are

expansions of $\text{HF}(I \cup N)$ adding all functions in δ and the constants v_1, \dots, v_k , where N is a structure whose universe is some suitably-chosen initial segment of the natural numbers, including zero, equipped with the binary relation symbol $\leq_{\mathbb{N}}$ with the obvious interpretation. We assume I and N to be disjoint.

For both kinds of program, the initial state S_0 interprets every dynamic function as the constant function mapping all inputs to \emptyset . It is a consequence of the finitary nature of the computation model that, in every subsequent state, any dynamic function will take values other than \emptyset for at most finitely many values of the arguments.

4.2 Terms, rules and programs

The *terms* over vocabulary (σ, δ) are defined as follows. We write $\llbracket t \rrbracket^S$ for the denotation of a term t in state S , which we do not define where it is obvious.

Variables. Every variable is a term.

Boolean constants. The constants false and true are terms denoting \emptyset and $\{\emptyset\}$, respectively.

Boolean combinations. If t_1 and t_2 are terms, then $\neg t_1$ and $t_1 \wedge t_2$ are terms. The Boolean connectives have the obvious denotation if their arguments take values true or false and denote false, otherwise.

Equality. If t_1 and t_2 are terms, then $t_1 = t_2$ is a term.

Ordering. If t_1 and t_2 are terms, then $t_1 \leq_{\mathbb{N}} t_2$ is a term. The denotation is obvious if both t_1 and t_2 denote numbers and false otherwise.

Set-theoretic functions. \emptyset and Atoms are terms; if t_1 and t_2 are terms, then $\cup t_1$, TheUnique(t_1), $t_1 \in t_2$ and $\{t_1, t_2\}$ are terms. Atoms denotes the set of atoms (i.e., $|I|$ or $|I| \cup N$) as appropriate) and TheUnique(a) denotes the unique element of a if it is a singleton set and denotes \emptyset , otherwise.

Counting. If t is a term, Card(t) is a term, denoting the cardinality of the set $\llbracket t \rrbracket^S$, modulo $\|N\|$, as an element of N , or 0 if t denotes an atom.³

Predicates. If $P \in \sigma$ is an n -ary relation symbol and t_1, \dots, t_n are terms, then $P(t_1, \dots, t_n)$ is a term. $\llbracket P(t_1, \dots, t_n) \rrbracket^S = \text{true}$ if $(\llbracket t_1 \rrbracket^S, \dots, \llbracket t_n \rrbracket^S) \in P^I$ and is false, otherwise. Note, in particular that, unless all the t_i denote atoms, the predicate term denotes false.

Dynamic functions. If $f \in \delta$ is an n -ary dynamic function and t_1, \dots, t_n are terms, then $f(t_1, \dots, t_n)$ is a term.

Comprehension. If v is a variable, $t(v)$, r and $g(v)$ are terms, with v not occurring free in r , then $T \equiv \{t(v) : v \in r : g(v)\}$ is a term, in which v is

³ In Section 4.4, we will introduce resource bounds for programs. The bounds will allow us to choose N to be sufficiently large with respect to $\|I\|$ that, in any program that operates within the resource bounds, the denotation of Card(t) is just $\|\llbracket t \rrbracket^S\|$ for any term t denoting a set.

bound. $\llbracket T \rrbracket = \{ \llbracket t \rrbracket^{S[a/v]} : a \in \llbracket r \rrbracket^S \text{ and } \llbracket g \rrbracket^{S[a/v]} = \text{true} \}$.

Note that our definition of states and terms differs slightly from the original definition of Blass et al. in [12]. They regard the set-theoretic elements as being constants and relations in the state; we include only \emptyset and \in as being part of the state as the other set-theoretic terms are easily defined from these.

For clarity, we will use the terms Numbers and Points to abbreviate the sets $\{ a : a \in \text{Atoms} : a \leq_{\mathbb{N}} a \}$ and $\{ a : a \in \text{Atoms} : \neg(a \leq_{\mathbb{N}} a) \}$, respectively.

The *rules* over vocabulary (σ, δ) are as follows. The denotation of a rule is the set of updates to the dynamic functions that it generates: these updates define the successor state. Write $\langle f, \bar{a}, b \rangle$ for the update that sets the value of $f(\bar{a})$ to b , where $f \in \delta$.

Skip. Skip is a rule. $\llbracket \text{Skip} \rrbracket^S = \emptyset$.

Update. If t_1, \dots, t_{n+1} are terms and $f \in \delta$ is an n -ary dynamic function, then $f(t_1, \dots, t_n) := t_{n+1}$ is a rule.

$$\llbracket f(\bar{t}) := t_{n+1} \rrbracket^S = \left\{ \left\langle f, \llbracket t_1 \rrbracket^S, \dots, \llbracket t_{n+1} \rrbracket^S \right\rangle \right\}.$$

Conditional. If t is a term and R_1 and R_2 are rules, **if t then R_1 else R_2 fi** is a rule with denotation $\llbracket R_1 \rrbracket^S$ if $\llbracket t \rrbracket^S = \text{true}$ and denotation $\llbracket R_2 \rrbracket^S$ otherwise.

Parallel execution. If v is a variable, t is a term in which v is not free and R is a rule, **do forall $v \in t$ R od** is a rule, in which v is bound.

$$\llbracket \text{do forall } v \in t \text{ } R \text{ od} \rrbracket = \bigcup \left\{ \llbracket R \rrbracket^{S[a/v]} : a \in \llbracket t \rrbracket^S \right\}.$$

A set of updates *clashes* if it contains $\langle f, \bar{a}, b \rangle$ and $\langle f, \bar{a}, b' \rangle$ for some $b \neq b'$, i.e., it tries to assign two distinct values to $f(\bar{a})$. If R is a rule and S a state, the *successor* of S is the state S' obtained by applying all the updates in $\llbracket R \rrbracket^S$ to S , unless $\llbracket R \rrbracket^S$ clashes, in which case $S' = S$.

A *program* is a *rule* without free variables. In fact, we can allow programs to be finite sets of rules without free variables, putting $\llbracket \{ R_1, \dots, R_n \} \rrbracket = \bigcup_i \llbracket R_i \rrbracket$. We can regard this as an abbreviation for the rule generated by recursively replacing $\{ R \}$ with R and $\{ R_1, \dots, R_n \}$ with the following, where v is a variable that does not occur free in any of the R_i .

```
do forall  $v \in \{ \text{true}, \text{false} \}$ 
  if  $v$  then  $\{ R_1, \dots, R_{\lfloor n/2 \rfloor} \}$  else  $\{ R_{\lfloor n/2 \rfloor + 1}, \dots, R_n \}$  fi
od
```

4.3 Runs

A *run* of a program is a finite or countable sequence of states $\langle S_i : i < \kappa \rangle$ for some $\kappa \leq \omega$, satisfying the following properties:

- S_0 interprets every dynamic function as the constant function with value \emptyset ;
- S_{i+1} is the successor of S_i for all i ;
- $\llbracket \text{Halt} \rrbracket^{S_i} = \text{true}$ only if $i + 1 = \kappa$.

If $\kappa = \omega$, we say that the run is non-terminating. Note that, if any stage produces a clashing update, the run must be non-terminating. In the following section, we introduce resource bounds for programs, which explain the possibility above that a run might be finite without its last state having $\text{Halt} = \text{true}$.

4.4 Polynomial bounds

In order to obtain the class $\tilde{\text{CPT}}(\text{Card})$, polynomial bounds are placed on both the number of stages for which a program is allowed to run and the number of objects within $\text{HF}(I \cup N)$ that it is allowed to use. Both bounds are necessary to guarantee that a program can be simulated in polynomial time by a Turing machine operating on a coding of the input structure. Bounding the number of stages is clearly necessary; bounding the number of objects ensures that only a polynomial number of parallel assignments is done at each stage, allowing the computation to be simulated in a polynomial number of steps on a sequential machine.

Call an object in $\text{HF}(I \cup N)$ *critical* at some stage of a run if it is true or false, is in the range of some dynamic function or is part of a tuple \bar{a} which some dynamic function maps to a value other than \emptyset . An object is *active* at a stage if it is in the transitive closure of some critical object.

Write $\text{Active}(I)$ for the substructure of $\text{HF}(I \cup N)$ containing exactly the objects that become active when a program Π is run on I . Because of the choiceless nature of the computation, $\text{Active}(I)$ is closed under all automorphisms of I . (Of course, N has no non-trivial automorphisms and there are no automorphisms exchanging numbers with elements of I as no element of I participates in $\leq_{\mathbb{N}}$.)

Definition 5 A $\tilde{\text{CPT}}(\text{Card})$ program of input vocabulary σ is a tuple $\bar{\Pi} = (\Pi, p, q)$, where Π is a program and $p, q \in \mathbb{N}$. The run of $\bar{\Pi}$ on a structure I is the greatest initial segment of the run of Π on I containing at most $\|I\|^p$ stages and $\|I\|^q$ active objects in total. For the states of the run, we take $|N| = \{0, \dots, \|I\|^q\}$.

Note that any program that activates a set x of size at least $\|I\|^q$ must activate more than $\|I\|^q$ objects (x and each of its members) and, thus, oversteps the resource bounds. Hence, in any program that operates within the bounds, N is large enough to represent the size of every set that is activated and the modular nature of the Card operator is moot. From this point, we will always assume that $|N| = \{0, \dots, \|I\|^q\}$.

We say that $\bar{\Pi}$ *accepts* input I if its run on that structure terminates in a state where Halt and Output are both true and *rejects* I if its run terminates in a state where Halt is true but Output is false. Notice that, if the program Π attempts to use more than its allocated resources of time or active objects, the run of $\bar{\Pi}$ will be a truncated version of the run of Π and will end in a state where Halt is false. Such a run neither accepts nor rejects.

A $\tilde{\text{CPT}}$ program is a $\tilde{\text{CPT}}(\text{Card})$ program containing no Card terms and no $\leq_{\mathbb{N}}$ terms. Blass et al. show in [12] that, for every $\tilde{\text{CPT}}$ program $\bar{\Pi}$, there is a Turing machine with polynomial time bounds that accepts exactly those strings that code structures accepted by $\bar{\Pi}$. This proof trivially extends to $\tilde{\text{CPT}}(\text{Card})$. Their main result is that $\tilde{\text{CPT}}$ is not the whole of \mathbf{P} because it cannot define the evenness query, which is easily defined in $\tilde{\text{CPT}}(\text{Card})$. On the other hand, $\tilde{\text{CPT}}$ does define all polynomial-time queries on ordered inputs.

Blass et al. also show in [12] that $\tilde{\text{CPT}}$ can define any polynomial-time property of small definable subsets $S \subseteq I$, where ‘small’ means that $\|S\|! \leq \|I\|$. This is because a program can define, in parallel, all $\|S\|!$ linear orders on S and use these to compute arbitrary polynomial-time queries on S . For instance, consider the family of structures consisting, for graphs G , of a copy of the graph \mathfrak{G}^0 or \mathfrak{G}^1 along with $\|\mathfrak{G}^0\|!$ isolated vertices. There is a $\tilde{\text{CPT}}$ program that distinguishes the padded versions of \mathfrak{G}^0 from the corresponding padded version of \mathfrak{G}^1 , a query that is not definable in $\text{IFP} + \text{C}$. In the following section, we improve on this result of Blass et al. by showing that $\tilde{\text{CPT}}$ can distinguish the the unpadded versions of \mathfrak{G}^0 and \mathfrak{G}^1 .

5 The algorithm

We now present a $\tilde{\text{CPT}}$ algorithm that determines the parity of ordered graphs \mathfrak{G}^T (that is, the parity of $\|T\|$). The algorithm does not involve counting but does use a slightly enriched model of computation. A $\tilde{\text{CPT}}$ program with input structure I ordinarily runs on $\text{HF}(I)$, the set of hereditarily finite sets over I ’s universe. For this section only, we enrich this universe with tuples and additional atoms 0 and 1.

Definition 6 Let I be a set. $\text{HF}^+(I)$ is the least set containing every element of $|I|$ along with 0 and 1, as atoms, and containing all its own finite subsets and all finite tuples of its elements.

In other words, $\text{HF}^+(I)$ treats tuples and the numbers 0 and 1 as first-class objects, rather than coding them as sets. We use the notation $\langle \dots \rangle$ for tuples of positive length in $\text{HF}^+(I)$ (we do not require the empty tuple). Because these new objects can be efficiently coded as sets, using this enriched universe does not affect the expressive power of programs. Objects in $\text{HF}^+(I)$ can be coded as sets in $\text{HF}(I)$ using a function such as

$$\begin{aligned} h(0) &= [0, 0] \\ h(1) &= [0, 1] \\ h(a) &= [1, a] && \text{for atoms } a \in I \\ h(x) &= [2, \{ h(y) : y \in x \}] && \text{for sets } x \in \text{HF}^+(I) \\ h(\langle x_1, \dots, x_n \rangle) &= [3, [h(x_1), [\dots, [h(x_n), \emptyset] \dots]]], \end{aligned}$$

where the numbers on the right-hand side are coded as von Neumann ordinals and $[x, y] = \{ x, \{ x, y \} \}$ is the standard Kuratowski coding of ordered pairs as sets.

The notion of rank extends to $\text{HF}^+(I)$ in the obvious way: \emptyset and the atoms have rank zero; tuples and non-empty sets have rank one higher than the greatest rank of their elements.

The algorithm for finding the parity of a CFI graph \mathfrak{G}^T proceeds as follows. The first step is to construct an object $\mu^T \in \text{HF}^+(\mathfrak{G}^T)$, that exhibits a high degree of symmetry. In particular, it is fixed not only by all automorphisms of \mathfrak{G}^T but also by all automorphisms of \mathfrak{G} — we call such objects *super-symmetric*. The key property of μ^T is that, despite its super-symmetry, $\mu^T = \mu^{T'}$ if, and only if, $\|T\| \equiv \|T'\| \pmod{2}$.

The next step is to transform μ^T into an object $B(\mu^T) \in \text{HF}^+(\emptyset)$ by replacing each pair of atoms e^0 and e^1 with the atoms 0 and 1. Because of the super-symmetry of μ^T , it does not matter whether we substitute $e^0 \mapsto 0$ and $e^1 \mapsto 1$ or $e^0 \mapsto 1$ and $e^1 \mapsto 0$, and the choice can be made for each e independently. The transformation retains the property that $B(\mu^T) = B(\mu^{T'})$ if, and only if, $\|T\| \equiv \|T'\| \pmod{2}$.

We now define a ‘parity function’ $p : \text{HF}^+(\emptyset) \rightarrow \{0, 1\}$, recursively as follows. Put $p(0) = 0$, $p(1) = 1$ and

$$\begin{aligned} p(\langle x_1, \dots, x_k \rangle) &\equiv \sum_i p(x_i) \pmod{2} \\ p(\{ x_1, \dots, x_k \}) &= \prod_i p(x_i). \end{aligned}$$

Note that the arithmetic required to compute $p(x)$ can be performed in $\tilde{\text{CPT}}$, without counting: for a set S , $p(S) = 0$ if, and only if, $p(s) = 0$ for some $s \in S$; the components of a tuple are ordered so we can compute the sum by inspecting the terms in turn.

The final step of the algorithm is to compute $p(B(\mu^T))$, which we show to be equal to the parity of $\|T\|$.

For the remainder of this section, fix a finite, connected, ordered graph $G = (V, E, \leq)$ of order n and let v_1, \dots, v_n enumerate V according to the linear order \leq . Recall that automorphisms of the graph \mathfrak{G} are given by ρ_H where $H \subseteq G$ and automorphisms of a Cai–Fürer–Immerman graph \mathfrak{G}^T are precisely those ρ_H where H is an even subgraph of G . Recall that each $\rho_H \in \text{Aut}(\mathfrak{G})$ naturally induces a bijection $\text{HF}^+(\hat{E}) \rightarrow \text{HF}^+(\hat{E})$ that extends the restriction of ρ_H to \hat{E} .

Definition 7 $x \in \text{HF}^+(\hat{E})$ is symmetric if it is fixed by all $\rho_H \in \text{Aut}(\mathfrak{G}^T)$ and super-symmetric if it is fixed by all $\rho_H \in \text{Aut}(\mathfrak{G})$.

Each vertex u of the form v_i^X is adjacent in \mathfrak{G} to exactly one of the vertices e^0 and e^1 for each edge e adjacent to v_i in G . Let $N(u)$ be the set of neighbours of u in \mathfrak{G} , and let $N_{\preceq}(u)$ be the tuple enumerating the elements of $N(u)$ according to the order \preceq . (Note that the restriction of \preceq to $N(u)$ is a linear order.)

Definition 8 For all $T \subseteq V$ and $i \in \{1, \dots, n\}$, let

$$\begin{aligned}\tau_i^T &= \left\{ N_{\preceq}(v_i^X) : X \subseteq E(v_i) : X \text{ is even} \iff v_i \notin T \right\} \\ \tilde{\tau}_i^T &= \left\{ N_{\preceq}(v_i^X) : X \subseteq E(v_i) : X \text{ is even} \iff v_i \in T \right\}.\end{aligned}$$

For example, if $E(v_i) = \{e, f, g\}$ where $e < f < g$, then

$$\begin{aligned}\tau_i^\emptyset &= \tilde{\tau}_i^{\{v_i\}} = \left\{ \langle e^0, f^0, g^0 \rangle, \langle e^1, f^1, g^0 \rangle, \langle e^1, f^0, g^1 \rangle, \langle e^0, f^1, g^1 \rangle \right\} \\ \tilde{\tau}_i^\emptyset &= \tau_i^{\{v_i\}} = \left\{ \langle e^1, f^0, g^0 \rangle, \langle e^0, f^1, g^0 \rangle, \langle e^0, f^0, g^1 \rangle, \langle e^1, f^1, g^1 \rangle \right\}.\end{aligned}$$

Lemma 9 For all $S, T \subseteq V$ and $k \in \{1, \dots, n\}$,

- (1) $\tau_k^S = \tau_k^T \iff \tilde{\tau}_k^S = \tilde{\tau}_k^T \iff v_k \notin S \Delta T$;
- (2) $\tau_k^S = \tilde{\tau}_k^T \iff \tilde{\tau}_k^S = \tau_k^T \iff v_k \in S \Delta T$.

PROOF. Immediate from definitions. \square

Definition 10 Let $\mu_1^T = \tau_1^T$ and $\tilde{\mu}_1^T = \tilde{\tau}_1^T$. For $i \in \{1, \dots, n-1\}$, let

$$\begin{aligned}\mu_{i+1}^T &= \left\{ \left\langle \mu_i^T, \tau_{i+1}^T \right\rangle, \left\langle \tilde{\mu}_i^T, \tilde{\tau}_{i+1}^T \right\rangle \right\} \\ \tilde{\mu}_{i+1}^T &= \left\{ \left\langle \mu_i^T, \tilde{\tau}_{i+1}^T \right\rangle, \left\langle \tilde{\mu}_i^T, \tau_{i+1}^T \right\rangle \right\}.\end{aligned}$$

Notice that $\tau_i^T, \tilde{\tau}_i^T, \mu_i^T, \tilde{\mu}_i^T \in \text{HF}^+(\hat{E})$ and $\tau_i^T \neq \tilde{\tau}_i^T$ and $\mu_i^T \neq \tilde{\mu}_i^T$ for all $T \subseteq V$ and $i \in \{1, \dots, n\}$.

Lemma 11 There is a $\tilde{\text{CPT}}$ program that, given input structure \mathfrak{G}^T , constructs the object μ_n^T .

PROOF. For each $i \in \{1, \dots, n\}$, the objects τ_i^T and $\tilde{\tau}_i^T$ can be constructed in $\text{deg}(v_i)$ steps. Therefore, we can construct maps (i.e. dynamic functions) $i \mapsto \tau_i^T$ and $i \mapsto \tilde{\tau}_i^T$ in $\mathcal{O}(n)$ steps. Maps $i \mapsto \mu_i^T$ and $i \mapsto \tilde{\mu}_i^T$ require an additional n steps to construct. The number of active objects is only $\mathcal{O}(\|\mathfrak{G}^T\|)$. \square

Note that, in constructing the sequence $\tau_1^T, \dots, \tau_n^T$, we are reliant on the linear order \leq on G 's vertices.

For $T \subseteq V$ and $k \in \{1, \dots, n\}$, write $T(k)$ for $T \cap \{v_1, \dots, v_k\}$.

Lemma 12 For all $S, T \subseteq V$ and $k \in \{1, \dots, n\}$,

$$\begin{aligned}(1) \quad \mu_k^S = \mu_k^T &\iff \tilde{\mu}_k^S = \tilde{\mu}_k^T \iff \|S(k)\| \equiv \|T(k)\| \pmod{2}; \\ (2) \quad \mu_k^S = \tilde{\mu}_k^T &\iff \tilde{\mu}_k^S = \mu_k^T \iff \|S(k)\| \not\equiv \|T(k)\| \pmod{2}.\end{aligned}$$

PROOF. Easy induction on k . \square

Corollary 13 For all $T \subseteq V$, μ_n^T is super-symmetric.

PROOF. Any $\rho_H \in \text{Aut}(\mathfrak{G})$ maps \mathfrak{G}^T to \mathfrak{G}^S where $S = T \Delta \text{odd}(H)$. ρ_H therefore maps μ_n^T to μ_n^S . Since $\|\text{odd}(H)\|$ is even and $\|T\| \equiv \|S\| \pmod{2}$, we have $\mu_n^T = \mu_n^S$. \square

For each $e^i \in \widehat{E}$, define the map $B_{e^i} : \text{HF}^+(\widehat{E}) \rightarrow \text{HF}^+(\widehat{E} \setminus \{e^0, e^1\})$ as follows:

$$B_{e^i}(x) = \begin{cases} 0 & \text{if } x = 0 \text{ or } e^i \\ 1 & \text{if } x = 1 \text{ or } e^{1-i} \\ x & \text{if } x \in \widehat{E} - \{e^0, e^1\} \\ \langle B_{e^i}(x_1), \dots, B_{e^i}(x_k) \rangle & \text{if } x = \langle x_1, \dots, x_k \rangle \\ \{ B_{e^i}(y) : y \in x \} & \text{if } x \text{ is a set.} \end{cases}$$

In other words, $B_{e^i}(x)$ is the function that replaces all instances of e^i with 0 and e^{1-i} with 1 throughout the transitive closure of x . Now, define the map $B : \text{HF}^+(\widehat{E}) \rightarrow \text{HF}^+(\emptyset)$ as

$$B(x) = B_{e_1^0}(B_{e_2^0}(\dots B_{e_m^0}(x) \dots)),$$

where e_1, \dots, e_m is the enumeration of E in the order induced by \preceq . We would like to compute $B(\mu_n^T)$ by a $\widetilde{\text{CPT}}$ algorithm. This calculation looks problematic, since e_i^0 and e_i^1 are indistinguishable in \mathfrak{G}^T up to isomorphism. We cannot compute $B(x)$ without also computing

$$\alpha(B(x)) = B_{\alpha(e_1^0)}(B_{\alpha(e_2^0)}(\dots B_{\alpha(e_m^0)}(x) \dots))$$

for each automorphism α of \mathfrak{G}^T . This would seem to require $\|\text{Aut}(\mathfrak{G}^T)\|$ active objects, which is not bounded by any polynomial in the size of \mathfrak{G}^T . The following lemma shows that the objects $B_{e_1^{i_1}}(B_{e_2^{i_2}}(\dots B_{e_m^{i_m}}(x) \dots))$ are, in fact, the same for any choice $i_1, \dots, i_m \in \{0, 1\}$.

Lemma 14 *If $x \in \text{HF}^+(\widehat{E})$ is super-symmetric, then $B_{e^0}(x) = B_{e^1}(x)$ and $B_{e^i}(x)$ is super-symmetric for every $e \in E$.*

PROOF. Let $x \in \text{HF}^+(\widehat{E})$ be super-symmetric, and let $e \in E$. Proof is by induction on the rank of x . The only super-symmetric objects of rank 0 are 0, 1 and \emptyset so the base case is trivial. For the inductive step, if $x = \langle x_1, \dots, x_n \rangle$, then each x_i is super-symmetric and the result is immediate from the inductive hypothesis. The only remaining case is x a set. Note that, for any y (not necessarily super-symmetric), $B_{e^i}(\rho_e(y)) = B_{e^{1-i}}(y)$ and that, since x is super-symmetric, $\rho_e(x) = x$. Therefore, $B_{e^0}(x) = B_{e^1}(\rho_e(x)) = B_{e^1}(x)$, as required.

To prove super-symmetry of $B_{e^i}(x)$, it suffices to show that ρ_f fixes $B_{e^i}(x)$ for all $f \in E$, since these automorphisms generate all of $\text{Aut}(\mathfrak{G})$. This is obvious when $f = e$, so we assume $f \neq e$. From the definition of B_{e^i} , it is clear that $\rho_f \circ B_{e^i} = B_{e^i} \circ \rho_f$. By the super-symmetry of x , we have $\rho_f(x) = x$. It follows that $\rho_f(B_{e^i}(x)) = B_{e^i}(\rho_f(x)) = B_{e^i}(x)$. \square

Lemma 15 *There is a $\tilde{\text{CPT}}$ program that, given input structure \mathfrak{G}^T , outputs the object $B(\mu_n^T)$.*

PROOF. We first use the $\tilde{\text{CPT}}$ program described in Lemma 11 to compute μ_n^T . We can then define a $\tilde{\text{CPT}}$ program which, given as input an object $x \in \text{HF}^+(\hat{E})$ and a distinguished atom e^i , computes $B_{e^i}(x)$ in $\mathcal{O}(\text{rank}(x))$ steps, using $\mathcal{O}(\|\text{TC}(x)\|)$ active objects. It is, therefore, possible to compute the sequence b_0, \dots, b_m in $\mathcal{O}(mn)$ additional steps, where $b_0 = \mu_n^T$ and $b_{i+1} = \text{TheUnique}(\{B_{e_{i+1}^0}(b_i), B_{e_{i+1}^1}(b_i)\})$. Finally, output b_m : by Lemma 14, $b_m = B(\mu_n^T)$. \square

Notice that $B(\mu_n^T) \in \text{HF}^+(\emptyset)$, so that $p(B(\mu_n^T)) \in \{0, 1\}$ is well-defined.

Lemma 16 $p(B(\mu_n^T)) \equiv \|T\| \pmod{2}$.

PROOF. By Lemma 12, it suffices to check the cases $T = \emptyset$ and $T = \{v_n\}$. For notational convenience, we set $P(x) = p(B(x))$.

Consider, first, the case $T = \emptyset$. For all $i \in \{1, \dots, n\}$, $P(\tau_i^\emptyset) = 0$, since each tuple in the set $B(\tau_i)$ contains an even number of 1's. Similarly, $P(\tilde{\tau}_i^\emptyset) = 1$, as each tuple in the set $B(\tilde{\tau}_i^T)$ contains an odd number of 1's.

We now show by induction that, for all $i \in \{1, \dots, n\}$, $P(\mu_i^\emptyset) = 0$ and $P(\tilde{\mu}_i^\emptyset) = 1$. The case $i = 1$ has already been dealt with, since $\mu_1 = \tau_1$ and $\tilde{\mu}_1 = \tilde{\tau}_1$. Suppose $P(\mu_i^\emptyset) = 0$ and $P(\tilde{\mu}_i^\emptyset) = 1$.

$$\begin{aligned} P(\mu_{i+1}^\emptyset) &= P(\{\langle \mu_i^\emptyset, \tau_{i+1}^\emptyset \rangle, \langle \tilde{\mu}_i^\emptyset, \tilde{\tau}_{i+1}^\emptyset \rangle\}) \\ &= (P(\mu_i^\emptyset) + P(\tau_{i+1}^\emptyset)) \cdot (P(\tilde{\mu}_i^\emptyset) + P(\tilde{\tau}_{i+1}^\emptyset)) \pmod{2} \\ &= (0 + 0) \cdot (1 + 1) \pmod{2} \\ &= 0. \end{aligned}$$

$$\begin{aligned} P(\tilde{\mu}_{i+1}^\emptyset) &= P(\{\langle \mu_i^\emptyset, \tilde{\tau}_{i+1}^\emptyset \rangle, \langle \tilde{\mu}_i^\emptyset, \tau_{i+1}^\emptyset \rangle\}) \\ &= (P(\mu_i^\emptyset) + P(\tilde{\tau}_{i+1}^\emptyset)) \cdot (P(\tilde{\mu}_i^\emptyset) + P(\tau_{i+1}^\emptyset)) \pmod{2} \\ &= (0 + 1) \cdot (1 + 0) \pmod{2} \\ &= 1. \end{aligned}$$

Therefore, $P(\mu_n^\emptyset) = 0$.

Now, let $T = \{v_n\}$. Similar reasoning shows that $P(\tau_i^T) = P(\mu_i^T) = 0$ and $P(\tilde{\tau}_i^T) = P(\tilde{\mu}_i^T) = 1$ for all $i \in \{1, \dots, n-1\}$. As $P(\tau_n^T) = 1$ and $P(\tilde{\tau}_n^T) = 0$,

we have

$$\begin{aligned}
P(\mu_n^T) &= P(\{ \langle \mu_{n-1}^T, \tau_n^T \rangle, \langle \tilde{\mu}_{n-1}^T, \tilde{\tau}_n^T \rangle \}) \\
&= (P(\mu_{n-1}^T) + P(\tau_n^T)) \cdot (P(\tilde{\mu}_{n-1}^T) + P(\tilde{\tau}_n^T)) \pmod{2} \\
&= (0 + 1) \cdot (1 + 0) \pmod{2} \\
&= 1. \qquad \square
\end{aligned}$$

Theorem 17 *There is a $\tilde{\text{CPT}}$ algorithm that, given input structure \mathfrak{G}^T , outputs $\|T\| \pmod{2}$.*

PROOF. Using Lemma 15, we first construct $B(\mu_n^T)$. It is easy to see that, for $x \in \text{HF}^+(\emptyset)$, $p(x)$ can be computed in $\mathcal{O}(\text{rank}(x))$ steps, while activating $\mathcal{O}(\|\text{TC}(x)\|)$ objects. This allows us to compute and output $p(B(\mu_n^T))$. Correctness is guaranteed by Lemma 16. \square

In the remainder of the paper, we revert to considering $\tilde{\text{CPT}}$ programs with input I as working on $\text{HF}(I)$ rather than $\text{HF}^+(I)$.

6 Fixed-point logics

In this section, we give some results on the fixed-point definability of queries defined by $\tilde{\text{CPT}}(\text{Card})$ programs and vice-versa. It is known from the work of Blass et al. [13] that $\text{IFP} + \text{C}$ cannot express every query definable in $\tilde{\text{CPT}}(\text{Card})$, and this also follows from the previous section of the present paper, since $\text{IFP} + \text{C}$ cannot determine the parity of CFI graphs.

However, we show that, for every $\tilde{\text{CPT}}(\text{Card})$ program $\bar{\Pi}$, there is an $\text{IFP} + \text{C}$ formula φ such that $\bar{\Pi}$ accepts I if, and only if, $\text{HF}(I \cup N) \models \varphi$. The proof is a relatively straightforward adaptation of the construction in [12] that translates a $\tilde{\text{CPT}}$ program on I to an IFP formula on $\text{HF}(I)$.

In this section, we use the notation $\alpha ? \beta : \gamma$ to abbreviate the formula $(\alpha \wedge \beta) \vee (\neg\alpha \wedge \gamma)$ and $\alpha?$ to abbreviate the formula $\alpha ? [\text{true}](x) : [\text{false}](x)$, where the subformulae are defined below.

Lemma 18 *For every state S and term t of appropriate vocabulary not mentioning the variable x , there is a formula $[t](x) \in \text{FO}(\#)$ such that $(S, a) \models [t]$ if, and only if, $a = \llbracket t \rrbracket^S$.*

PROOF. By induction on the structure of t .

$$\begin{aligned}
[\text{false}](x) &\equiv x = \emptyset \\
[\text{true}](x) &\equiv \forall y (y \in x \leftrightarrow y = \emptyset) \\
[v_i](x) &\equiv x = v_i \\
[\emptyset](x) &\equiv x = \emptyset \\
[\text{Atoms}](x) &\equiv \forall u (u \in x \leftrightarrow (u \neq \emptyset \wedge \forall v v \notin u)) \\
[\bigcup t](x) &\equiv \forall u (u \in x \leftrightarrow \exists v w ([t](w) \wedge u \in v \in w)) \\
[\text{TheUnique}(t)](x) &\equiv \exists u ([t](u) \wedge \exists v \forall w (w \in u \leftrightarrow w = v)) ? \\
&\quad \exists u ([t](u) \wedge x \in u) : x = \emptyset \\
[t_1 \in t_2](x) &\equiv (\forall uv (([t_1](u) \wedge [t_2](v)) \rightarrow u \in v)) ? \\
[\{t_1, t_2\}](x) &\equiv \forall u (u \in x \leftrightarrow ([t_1](u) \vee [t_2](u))) \\
[\text{Card}(t)] &\equiv \exists u ([t](u) \wedge \#v (v \in u) = x) \\
[\neg t_1](x) &\equiv [t_1](\emptyset) ? \\
[t_1 \wedge t_2](x) &\equiv (\exists u ([\text{true}](u) \wedge [t_1](u) \wedge [t_2](u))) ? \\
[P(t_1 \dots t_n)](x) &\equiv (\exists u_1 \dots u_n (P(u_1 \dots u_n) \wedge \bigwedge_i [t_i](u_i))) ? \\
[f(t_1 \dots t_n)](x) &\equiv \exists u_1 \dots u_n (f(u_1 \dots u_n) = x \wedge \bigwedge_i [t_i](u_i)) \\
[\{t(v) : v \in r : g(v)\}](x) &\equiv \forall u (u \in x \leftrightarrow \exists v ([t(v)](u) \wedge \exists w ([r](w) \wedge v \in w) \\
&\quad \wedge \exists w ([g(v)](w) \wedge [\text{true}](w))) \\
&\quad \wedge ((\forall u u \notin x) \leftrightarrow x = \emptyset).
\end{aligned}$$

The terms $t_1 = t_2$ and $t_1 \leq_{\mathbb{N}} t_2$ can be dealt with in the same way as predicate terms $P(t_1, \dots, t_n)$. \square

Note that counting is required only for the translation of Card terms.

Lemma 19 *Let R be a rule of vocabulary (σ, δ) , let f be a function symbol in δ and let S be a state. There is a formula $\nu_{R,f}(\bar{x}y) \in \text{FO}(\#)$ such that $(S, \bar{a}, b) \models \nu_{R,f}$ if, and only if, $\langle f, \bar{a}, b \rangle \in \llbracket R \rrbracket^S$ and $\llbracket R \rrbracket^S$ is non-clashing.*

PROOF. Suppose we have a formula $\nu'_{R,f}(\bar{x}y)$ such that $(S, \bar{a}, b) \models \nu_{R,f}$ if, and only if, $\langle f, \bar{a}, b \rangle \in \llbracket R \rrbracket^S$. That is, ν' doesn't worry about clashes. We can then put

$$\nu_{R,f}(\bar{x}y) \equiv \nu'_{R,f}(\bar{x}y) \wedge \bigwedge_{g \in \delta} \forall \bar{u} v w ((\nu'_{R,g}(\bar{u}v) \wedge \nu'_{R,g}(\bar{u}w)) \rightarrow v = w).$$

We now define ν' by induction on the structure of R . If R is **Skip** or $g(\bar{t}) := t$ for some $g \neq f$, then $\nu'_{R,f} \equiv \text{false}$. Otherwise, for $R \equiv f(t_1 \dots t_n) := t_{n+1}$,

$$\nu'_{R,f}(\bar{x}y) \equiv [t_{n+1}](y) \wedge \bigwedge_{1 \leq i \leq n} [t_i](x_i),$$

for $R \equiv \text{if } t \text{ then } R_1 \text{ else } R_2 \text{ fi}$,

$$\nu'_{R,f}(\bar{x}y) \equiv \exists u \left([t](u) \wedge [\text{true}](u) \right) ? \nu'_{R_1,f}(\bar{x}y) : \nu'_{R_2,f}(\bar{x}y)$$

and, for $R \equiv \text{do forall } v \in t \text{ } R_1 \text{ od}$,

$$\nu'_{R,f}(\bar{x}y) \equiv \exists uw \left(u \in w \wedge [t](w) \wedge \nu'_{R(u),f}(\bar{x}y) \right). \quad \square$$

Theorem 20 *Let $\bar{\Pi} = (\Pi, p, q)$ be a program of vocabulary (σ, δ) , with input I . There is a formula $\varphi \in \text{IFP} + \text{C}$ such that $\text{HF}(I \cup N) \models \varphi$ if, and only if, $\bar{\Pi}$ accepts I .*

PROOF. We show that relations D_f for each $f \in \delta$ are simultaneously definable in $\text{IFP} + \text{C}$, where $(i, \bar{x}, y) \in D_f$ if, and only if, $f(\bar{x}) = y \neq \emptyset$ at stage i .

We construct the D_f stage by stage. Let $\chi(i)$ denote the formula that says that i is the current stage: that is, $\chi(i)$ asserts that i is the least number not yet appearing as the first component of any tuple in any D_f . Let $\nu_{\bar{\Pi},f}^*$ be $\nu_{\bar{\Pi},f}$ with every subformula $g(\bar{u}) = v$ replaced by

$$D_g(i-1, \bar{u}, v) \vee \left(v = \emptyset \wedge \forall w \neg D_g(i-1, \bar{u}, w) \right).$$

That is, $\nu_{\bar{\Pi},f}^*$ is a version of $\nu_{\bar{\Pi},f}$ that obtains the value of the dynamic functions from the relations D_g instead of from the structure representing the current state.

The new value of D_f is defined by

$$\varphi_f(i, \bar{x}, y) \equiv \chi(i) \wedge y \neq \emptyset \wedge \left[\nu_{\bar{\Pi},f}^*(\bar{x}y) \vee \left(D_f(i-1, \bar{x}, y) \wedge \forall z \neg \nu_{\bar{\Pi},f}^*(\bar{x}z) \right) \right],$$

where $i-1$ denotes the immediate $\leq_{\mathbb{N}}$ -predecessor of i (or 0 if $i = 0$). φ_f says that, either $f(\bar{x})$ is set to $y \neq \emptyset$ at stage i or it keeps its value from the previous stage. Once the D_f are defined, it is straightforward to write the required formula, using counting to simulate Card terms and to check that the resource bounds are not exceeded.⁴ \square

⁴ We do not actually need to use counting for this result. Instead, we could check

In fact, φ does not need access to the whole of $\text{HF}(I \cup N)$ but only needs the elements of $\text{Active}(I)$ along with the numbers up to the greater of $\|I\|^p$ and $\|I\|^q$ to allow the numbering of the stages and counting of any set of objects that may become active. Call this structure $\text{Active}^+(I)$.

Corollary 21 *Let $\bar{\Pi} = (\Pi, p, q)$ be a program with input I . There is a formula $\varphi \in \text{IFP} + \text{C}$ such that $\text{Active}^+(I) \models \varphi$ if, and only if, $\bar{\Pi}$ accepts I .*

We can also simulate fixed-point formulae with $\tilde{\text{CPT}}(\text{Card})$ programs.

Theorem 22 *Let $\varphi(\bar{u}) \in \text{IFP} + \text{C}$. There is a program $\bar{\Pi}$ of $\tilde{\text{CPT}}(\text{Card})$ that, on input (I, \bar{a}) , determines whether $(I, \bar{a}) \models \varphi$. Further, $\bar{\Pi}$ uses only sets of rank at most 1 and uses counting if, and only if, φ does.*

PROOF. It follows from [19] that we may assume φ to be of the form $\exists \lambda \exists y (\mathbf{ifp}_{X, \bar{x}} \psi) (\lambda \dots \lambda y \dots y)$, where $\psi \in \text{FO}(\#)$, λ is a number variable and y a point variable. We may also assume that every quantifier in ψ uses a fresh variable.

Let $\bar{z} = z_1 \dots z_r$ enumerate the variables of φ and let ψ_1, \dots, ψ_n enumerate the subformulae of ψ , with $\psi_n \equiv \psi$ and $i < j$ whenever ψ_i is a proper subformula of ψ_j . For each ψ_i , the program will build up a dynamic function $f_i(\bar{z})$ such that, after simulating t stages of the fixed-point, $f_i(a_1 \dots a_r) = a_1$ if $(I, X_t, \bar{a}) \models \psi_i$ and $f_i(\bar{a}) = \emptyset$, otherwise. In particular, note that $a_1 \in |I|$, so $a_1 \neq \emptyset$, and that some projection of f_n defines X .

For each ψ_i , we write a rule $R_i(\bar{z})$ that updates f_i to define the tuples that satisfy ψ_i , so long as each f_j with $j < i$ defines the tuples that satisfy ψ_j at the current stage. This is trivial for atomic formulae and Boolean combinations; the case $\psi_i \equiv \exists z_k \psi_j$, can be dealt with by checking whether the set $\{a : a \in S : f_j(\bar{z}[a/z_k]) \neq \emptyset\}$ is empty, where S is either Points or Numbers according to the sort of z_k .

We can use a sequence of nullary dynamic functions to ensure that the R_i are fired in sequence rather than in parallel, and further rules to detect whether a fixed-point has been reached and to set Output and Halt appropriately. This

that $\|S\| = n$ by asserting the existence of an object in $\text{HF}(I \cup N)$ that codes a bijection between the sets S and $\{0, \dots, n-1\}$ and count the number of stages and active objects in a similar manner. However, these bijections are not, in general, activated by the program being simulated so will not be available when we restrict to active objects for Corollary 21.

can then be wrapped in nested rules of the form

```

do forall  $z_1 \in S_1$ 
  do forall  $z_2 \in S_2$ 
    ...
  od
od,

```

where each S_i is either Points or Numbers, as before. Finally, add rules to ‘remember’ the value of X from the previous iteration of the induction to determine when the fixed-point has been reached.

It is easily checked that the program described evaluates φ in a polynomial number of steps, activating at most polynomially many objects, none of which has rank greater than 1. \square

7 Supports

Let G be a group acting on a set I . The action of G on I extends naturally to an action on $\text{HF}(I)$ defined inductively by putting $g(x) = \{g(y) : y \in x\}$ for every set $x \in \text{HF}(I)$ and $g \in G$.

For all $x \in \text{HF}(I)$, let $\text{Stab}_G(x) = \{g \in G : g(x) = x\}$ denote the stabilizer subgroup of x . If x is a set, let $\text{Stab}_G^\bullet(x) = \bigcap_{y \in x} \text{Stab}_G(y)$ denote the pointwise stabilizer of x . Note that $\text{Stab}_G^\bullet(x) \subseteq \text{Stab}_G(x)$ for every set x . For all $x \in \text{HF}(I)$, let $\text{Orbit}(x) = \{g(x) : g \in G\}$ and for every $x \in \text{HF}(I)$, let $\mu(x) = \max_{y \in \text{TC}(x)} \|\text{Orbit}(y)\|$.

For all $x, x' \in \text{HF}(I)$, write $x \sim_G x'$ if $x' \in \text{Orbit}(x)$. Note that \sim_G is an equivalence relation on $\text{HF}(I)$. A set x is *connected* if $y \sim_G y'$ for all $y, y' \in x$. The *connected components* of a set are its maximal connected subsets. Note that a non-empty set x is connected precisely when it has exactly one connected component, namely x itself.

A subset $S \subseteq I$ is a *support* for x if $\text{Stab}_G^\bullet(S) \subseteq \text{Stab}_G(x)$; that is, every $g \in G$ that fixes S pointwise also fixes x (but not necessarily pointwise). Let $\sigma(x)$ denote the minimal size of a support for x . For all $r, m \in \mathbb{N}$, let

$$\sigma_{\max}(r, m) = \max \{ \sigma(x) : x \in \text{HF}(I), \text{rank}(x) \leq r \text{ and } \mu(x) \leq m \}.$$

The main theorem of this section is:

Theorem 23 *If G is Abelian, then $\sigma_{\max}(r, m) \leq (\log_2 m)^r$ for all $r, m \in \mathbb{N}$.*

As a corollary of Theorem 23, we obtain a bound on the size of supports of active elements for $\tilde{\text{CPT}}(\text{Card})$ programs over CFI graphs.

The upper bound in Theorem 23 can be shown to be tight for Abelian groups. In the non-Abelian setting, we do not know whether there exists an upper bound on the function $\sigma_{\max}(r, m)$ in terms of r and m alone (excluding the size of I). This might be an interesting question relevant to the study of $\tilde{\text{CPT}}(\text{Card})$.

Lemma 24 *Let x_1, \dots, x_n be the connected components of a set x . Then $\text{Stab}_G(x) = \text{Stab}_G(x_1) \cap \dots \cap \text{Stab}_G(x_n)$.*

PROOF. The inclusion $\text{Stab}_G(x_1) \cap \dots \cap \text{Stab}_G(x_n) \subseteq \text{Stab}_G(x)$ is obvious, since every element of G that fixes each component of x also fixes x . For the converse, consider $g \in \text{Stab}_G(x)$ and, towards a contradiction, assume $g \notin \text{Stab}_G(x_1) \cap \dots \cap \text{Stab}_G(x_n)$. There must be some $i \in \{1, \dots, n\}$ such that $g \notin \text{Stab}_G(x_i)$ and there must be some $y \in x_i$ such that $g(y) \notin x_i$. Since $g \in \text{Stab}_G(x)$, it follows that $g(y) \in x$ so there exists $j \neq i$ such that $g(y) \in x_j$. But this yields the desired contradiction, since $y \sim_G g(y)$ yet $y_i \not\sim_G y_j$ for all $y_i \in x_i$ and $y_j \in x_j$. \square

Lemma 25 *Let x' be a connected component of a set x . Then $\mu(x') \leq \mu(x)$.*

PROOF. $\text{Stab}_G(x) \subseteq \text{Stab}_G(x')$ by Lemma 24. By the orbit–stabilizer theorem, we have $\|\text{Orbit}(x')\| \leq \|\text{Orbit}(x)\|$. Since $\text{TC}(x') \setminus \{x'\} \subseteq \text{TC}(x)$, we have,

$$\begin{aligned} \mu(x') &= \max \left\{ \|\text{Orbit}(x')\|, \max_{y \in \text{TC}(x') \setminus \{x'\}} \|\text{Orbit}(y)\| \right\} \\ &\leq \max \left\{ \|\text{Orbit}(x)\|, \max_{y \in \text{TC}(x)} \|\text{Orbit}(y)\| \right\} \\ &= \mu(x). \end{aligned} \quad \square$$

Lemma 26 *For Abelian G , if $x \sim_G y$ then $\text{Stab}_G(x) = \text{Stab}_G(y)$.*

PROOF. Suppose $y = g(x)$. For all $h \in G$, notice that

$$h(x) = x \iff gh(x) = g(x) \iff hg(x) = g(x) \iff h(y) = y.$$

So exactly the same elements of G fix x as fix y . Therefore, $\text{Stab}_G(x) = \text{Stab}_G(y)$. \square

Lemma 27 *Let G be Abelian and let x be a connected set with $y \in x$. Then $\text{Stab}_G(y) \subseteq \text{Stab}_G(x)$.*

PROOF. For every $y' \in x$, Lemma 26 implies that $\text{Stab}_G(y) = \text{Stab}_G(y')$, as $y \sim_G y'$ by connectedness of x . Thus, we have $\text{Stab}_G(y) = \bigcap_{y' \in x} \text{Stab}_G(y') = \text{Stab}_G^\bullet(x) \subseteq \text{Stab}_G(x)$. \square

Lemma 28 *If G is Abelian and x is a non-empty connected set, then $\sigma(x) \leq \sigma_{\max}(\text{rank}(x) - 1, \mu(x))$.*

PROOF. Suppose x is a non-empty connected set and $y \in x$. By Lemma 27, $\text{Stab}_G(y) \subseteq \text{Stab}_G(x)$. Therefore, $\sigma(x) \leq \sigma(y)$ since any subset of I that supports y also supports x . Since $\mu(y) \leq \mu(x)$ and $\text{rank}(y) = \text{rank}(x) - 1$, we have

$$\sigma(x) \leq \sigma(y) \leq \sigma_{\max}(\text{rank}(y), \mu(y)) \leq \sigma_{\max}(\text{rank}(x) - 1, \mu(x)). \quad \square$$

Lemma 29 *Let x_1, \dots, x_n be the connected components of a set x , let $y_0 = \emptyset$ and $y_j = x_1 \cup \dots \cup x_j$ for all $j \in \{1, \dots, n\}$, and let*

$$J = \{j : \text{Stab}_G(y_j) \neq \text{Stab}_G(y_{j-1})\}.$$

Then $\|J\| \leq \log_2 \|\text{Orbit}(x)\|$ and $\sigma(x) \leq \sum_{j \in J} \sigma(x_j)$.

PROOF. Let $j_0 = 0$ and let $J = \{j_1, \dots, j_t\}$ where $1 \leq j_1 < \dots < j_t \leq n$. Note that

$$\begin{aligned} G &= \text{Stab}_G(y_{j_0}) \supsetneq \text{Stab}_G(y_{j_1}) \supsetneq \text{Stab}_G(y_{j_2}) \supsetneq \dots \supsetneq \text{Stab}_G(y_{j_{t-1}}) \\ &\supsetneq \text{Stab}_G(y_{j_t}) = \text{Stab}_G(x_1) \cap \dots \cap \text{Stab}_G(x_n) = \text{Stab}_G(x), \end{aligned}$$

where the last equality is by Lemma 24. By the orbit–stabilizer theorem, we have that

$$\|\text{Orbit}(x)\| = \frac{\|G\|}{\|\text{Stab}_G(x)\|} = \prod_{i=1}^t \frac{\|\text{Stab}_G(y_{j_{i-1}})\|}{\|\text{Stab}_G(y_{j_i})\|} \geq 2^t.$$

Therefore, $\|J\| = t \leq \log_2 \|\text{Orbit}(x)\|$.

It remains to prove the bound on $\sigma(x)$. For all $j \in J$, let $S_j \subseteq I$ be a support of least cardinality for x_j , so that $\|S_j\| = \sigma(x_j)$. Let $S = \bigcup_{j \in J} S_j$. Notice that $\text{Stab}_G^\bullet(S) = \bigcap_{j \in J} \text{Stab}_G^\bullet(S_j) \subseteq \bigcap_{j \in J} \text{Stab}_G(x_j) = \text{Stab}_G(x)$. Thus, S supports x and so $\sigma(x) \leq \|S\| \leq \sum_{j \in J} \sigma(x_j)$, as claimed. \square

We are now ready to prove the main theorem, that $\sigma_{\max}(r, m) \leq (\log_2 m)^r$ for all $r, m \in \mathbb{N}$. That is, every object $x \in \text{HF}(I)$ with rank at most r and with $\|\text{Orbit}(y)\| \leq m$ for every $y \in \text{TC}(x)$, has a support of size at most $(\log_2 m)^r$.

Proof of Theorem 23. By induction on r . For the base case, the only objects of rank 0 are the atoms and \emptyset . Each atom a has a support of size 1 (namely, $\{a\}$) and \emptyset has empty support. Therefore, $\sigma_{\max}(0, m) = 1$, as required.

Let $r \in \mathbb{N}$ and assume that $\sigma_{\max}(r, m) \leq (\log_2 m)^r$ for all $m \in \mathbb{N}$. We now argue that $\sigma_{\max}(r+1, m) \leq (\log_2 m)^{r+1}$. Consider $x \in \text{HF}(I)$ with $\text{rank}(x) = r+1$ and let $m = \mu(x)$. Let x_1, \dots, x_n be the connected components of x and let $J \subseteq \{1, \dots, n\}$ be as in Lemma 29. That lemma implies that $\|J\| \leq \log_2 \|\text{Orbit}(x)\| \leq \log_2 m$ and that $\sigma(x) \leq \sum_{j \in J} \sigma(x_j)$. Each x_i is a non-empty connected set with $\text{rank}(x_i) \leq r+1$ and $\mu(x_i) \leq m$ by Lemma 25. By Lemma 28 and the inductive hypothesis, we have $\sigma(x_j) \leq \sigma_{\max}(r, m) \leq (\log_2 m)^r$. As this holds for all $j \in J$, we have

$$\sigma(x) \leq \sum_{j \in J} \sigma(x_j) \leq \|J\| \cdot \sigma_{\max}(r, m) \leq \log_2 m \cdot (\log_2 m)^r,$$

as required. \square

Now, let $\bar{\Pi}$ be a $\tilde{\text{CPT}}(\text{Card})$ program with an active element bound of n^q for some $q > 0$. Blass et al. have shown that that, because the computation is choiceless, if $\bar{\Pi}$ activates an object $x \in \text{HF}(I)$, then it also activates every object in the orbit of x under $\text{Aut}(I)$, as well as every object in $\text{TC}(x)$. Therefore, we must have $\mu(x) \leq \|I\|^q$ for every object x activated by $\bar{\Pi}$ over I . This observation combined with Theorem 23 yields:

Corollary 30 *Let G be an Abelian subgroup of $\text{Aut}(I)$. Every object x activated in $\bar{\Pi}$'s run on I has a support of size at most $(q \log_2 \|I\|)^{\text{rank}(x)}$ with respect to the action of G .*

We will use supports to show that, if a $\tilde{\text{CPT}}(\text{Card})$ program activates objects only of rank at most some fixed r , independent of its input, then it cannot determine the parity of CFI graphs. It is not hard to see that, for any ordered graph $G = (V, E, \leq)$, the pre-ordered CFI graphs \mathfrak{G}^0 and \mathfrak{G}^1 have the same Abelian automorphism group. Also note that, if we fix some k and restrict attention to k -regular graphs G , we have $\|\mathfrak{G}^0\| = \mathcal{O}(\|G\|)$.

Corollary 31 *Let $\bar{\Pi}$ be a $\tilde{\text{CPT}}(\text{Card})$ program that activates objects of rank at most r . For every fixed k , there is a constant c such that, for any sufficiently large ordered, k -regular graph G , every object activated in the run of $\bar{\Pi}$ on \mathfrak{G}^0 or \mathfrak{G}^1 has a support of size $\mathcal{O}((c \log_2 \|G\|)^r)$ with respect to the action of $\text{Aut}(\mathfrak{G}^0)$.*

8 Equivalence

Given a structure I of vocabulary σ and a constant k , the *transitively k -supported* elements of $\text{HF}(I)$ are those elements x where every element of x 's transitive closure has a support of size at most k .⁵ We abuse notation and write \bar{I}_k for both the transitively k -supported part of $\text{HF}(I)$ and the corresponding structure of vocabulary $\langle \sigma, \in, \emptyset \rangle$.

The proof of Blass et al. in [12] that evenness is not definable in $\tilde{\text{CPT}}$ has a counterpart to our Corollary 31 showing that there is a constant bound on the size of supports of active elements when the input is an unstructured set. This is then combined with a statement showing that, if I and J are sets which cannot be distinguished in \mathcal{L}^{mk} , then $\bar{I}_k \equiv^{\mathcal{L}^m} \bar{J}_k$. We generalize this in Theorem 33 in two ways. We consider the counting logics \mathcal{C}^m and \mathcal{C}^{mk} and we relax the hypothesis from requiring I and J to be unstructured sets, allowing any pair of \mathcal{C}^{mk} -homogeneous structures (defined below).

Recall that the \mathcal{C}^m -type of a tuple \bar{a} in a structure I is the collection of \mathcal{C}^m formulae that are true in (I, \bar{a}) .

Definition 32 *A structure I is \mathcal{C}^m -homogeneous if, whenever \bar{a} and \bar{b} have the same \mathcal{C}^m -type in I , there is an automorphism of I that maps \bar{a} to \bar{b} .*

In particular, every ordered CFI graph \mathfrak{G}^0 or \mathfrak{G}^1 is \mathcal{C}^n -homogeneous for any $n \geq 2$ as there is an automorphism mapping a to b if, and only if, the two elements are in the same equivalence class of the pre-order. Any finite linear order, such as N , is \mathcal{C}^n -homogeneous for any $n \geq 2$. In the case where a structure I is to be used as input to a $\tilde{\text{CPT}}(\text{Card})$ program, we may assume that I contains a linear order of sufficient length to number the stages of the computation and to denote any numbers required by Card terms.

We can now state the main theorem of this section.

Theorem 33 *Let $k \geq 0$ and $m > 1$ and let I and J be \mathcal{C}^{mk} -homogeneous structures of the same vocabulary. If $I \equiv^{\mathcal{C}^{mk}} J$ then $\bar{I}_k \equiv^{\mathcal{C}^m} \bar{J}_k$.*

Towards a proof of the theorem, fix appropriate k , m , I and J . We may assume $k > 0$ as the result is trivial otherwise.

A *molecule* on a structure I (or, more succinctly, an I -molecule) is a sequence $\alpha = \alpha_1 \dots \alpha_k$ of atoms. (We relax the condition of Blass et al. that the atoms

⁵ Blass et al. use the term ‘ k -symmetric’ for the transitively k -supported objects; our term is, we feel, more descriptive and avoids confusion with the ‘symmetric’ and ‘super-symmetric’ objects of Section 5.

must be distinct.) The intention is to use molecules as supports for sets.

The m -ary *type* of a sequence $\bar{\alpha} = \alpha_1 \dots \alpha_m$ of molecules is just the \mathcal{C}^{mk} -type of the sequence of atoms $\alpha_{11} \dots \alpha_{mk}$ in I . Write $\text{tp}_I(\bar{\alpha})$ for the type of $\bar{\alpha}$. The following lemma is immediate from the definition of types and the fact that \mathcal{C}^{mk} -types are determined by a single formula on finite structures [16].

Lemma 34 *For some $\ell < m$, suppose that $\alpha_0 \dots \alpha_\ell$ are I -molecules and that $\beta_1, \dots, \beta_\ell$ are J -molecules. If $\text{tp}_J(\beta_1, \dots, \beta_\ell) = \text{tp}_I(\alpha_1, \dots, \alpha_\ell)$, there is a J -molecule β_0 such that $\text{tp}_J(\beta_0, \dots, \beta_\ell) = \text{tp}_I(\alpha_0, \dots, \alpha_\ell)$.*

The definition of *forms* is adapted from the corresponding definition in [12]. Forms can be thought of as templates for building transitively k -supported sets from molecules.

Definition 35 *Fix a list $c_1 \dots c_k$ of new symbols. The set of forms is the least set containing each of the c_i and every finite set of pairs (φ, τ) , where φ is a form and τ a binary type.*

The *rank* of a form is defined inductively: $\text{rank } c_i = 0$, for each i , and if φ is a set, $\text{rank } \varphi = 1 + \max \{ \text{rank } \psi : (\psi, \tau) \in \varphi \}$.

Definition 36 *The denotation of a form φ and a molecule α over a structure I is $\varphi \star \alpha \in \text{HF}(I)$, where $c_i \star \alpha = \alpha_i$ and $\varphi \star \alpha = \{ \psi \star \beta : (\psi, \text{tp}_I(\alpha, \beta)) \in \varphi \}$ if φ is a set.*

We want to show that the set of denotations of forms and I -molecules corresponds exactly to \bar{I}_k . We first prove a useful result on the action of automorphisms on denotations.

Lemma 37 *If $\rho \in \text{Aut}(I)$ then $\rho(\varphi \star \alpha) = \varphi \star \rho(\alpha)$.*

PROOF. By induction on the structure of φ . If $\varphi = c_i$, the result is trivial so suppose φ is a set.

$$\begin{aligned} \rho(\varphi \star \alpha) &= \{ \rho(\psi \star \beta) : (\psi, \text{tp}_I(\alpha, \beta)) \in \varphi \} \\ &= \{ \psi \star \rho(\beta) : (\psi, \text{tp}_I(\alpha, \beta)) \in \varphi \} \\ &= \{ \psi \star \rho(\beta) : (\psi, \text{tp}_I(\rho(\alpha), \rho(\beta))) \in \varphi \} \\ &= \varphi \star \rho(\alpha). \end{aligned}$$

The second line is by the inductive hypothesis; the third follows because the types of molecules are preserved under automorphisms. \square

Lemma 38 *$x \in \bar{I}_k$ if, and only if, $x = \varphi \star \alpha$ for some I -molecule α and some form φ .*

PROOF. (\Leftarrow) Suppose $x = \varphi \star \alpha$ and let ρ be an automorphism of I that fixes α . By Lemma 37, $\rho(\varphi \star \alpha) = \varphi \star \rho(\alpha) = \varphi \star \alpha$. Therefore, α supports $\varphi \star \alpha$. Further, every $y \in \text{TC}(x)$ is also of the form $\psi \star \beta$ so also has a support of size k . Hence, $x \in \bar{I}_k$.

(\Rightarrow) By induction on the structure of x . If $x = a$ for some atom a , then $x = c_1 \star \alpha$ for any molecule α with $\alpha_1 = a$. If $x = \emptyset$, then $x = \emptyset \star \alpha$ for any α .

Suppose that $x \in \bar{I}_k$ is a non-empty set. There must be some molecule α supporting x . By the inductive hypothesis, $x = \{ \varphi_y \star \alpha_y : y \in x \}$. We show that $x = \varphi \star \alpha$, where $\varphi = \{ (\varphi_y, \text{tp}_I(\alpha, \alpha_y)) : y \in x \}$.

$$\begin{aligned} \varphi \star \alpha &= \{ \psi \star \beta : (\psi, \text{tp}_I(\alpha, \beta)) \in \varphi \} \\ &= \{ \varphi_y \star \beta : y \in x \text{ and } \text{tp}_I(\alpha, \beta) = \text{tp}_I(\alpha, \alpha_y) \} \end{aligned}$$

Clearly, then, $x \subseteq \varphi \star \alpha$. For the converse, suppose $z \in \varphi \star \alpha$. We must have $z = \psi \star \beta$ with $(\psi, \text{tp}_I(\alpha, \beta)) \in \varphi$. Further, there must be $y \in x$ with $\psi = \varphi_y$ and $\text{tp}_I(\alpha, \beta) = \text{tp}_I(\alpha, \alpha_y)$. It follows from the \mathcal{C}^{mk} -homogeneity of I that there is an automorphism ρ of I such that $\rho(\alpha\beta) = \alpha\alpha_y$. Since ρ fixes α pointwise, it also fixes x so $z \in x$. \square

The final pieces of machinery we need in order to prove Theorem 33 are the relations Eq and In. These allow us to determine whether $\varphi \star \alpha =$ (resp., \in) $\psi \star \beta$ by considering only the forms φ and ψ and the type of α and β , independent of the structure from which the molecules come.

Lemma 39 *There are ternary relations Eq and In such that, for all \bar{I}_k , all forms φ and ψ and molecules α and β ,*

$$\begin{aligned} \varphi \star \alpha &= \psi \star \beta \iff \text{Eq}(\varphi, \psi, \text{tp}(\alpha, \beta)) \\ \varphi \star \alpha &\in \psi \star \beta \iff \text{In}(\varphi, \psi, \text{tp}(\alpha, \beta)) \end{aligned}$$

PROOF. Define Eq and In as follows, where τ is any realizable binary type.

$$\begin{aligned} \text{Eq}(\varphi, \psi, \tau) &\iff \\ &\left[\varphi = c_i, \psi = c_j \text{ and } (\alpha_i = \beta_j) \in \tau \right] \text{ or} \\ &\left[\varphi, \psi \text{ are sets and, for all forms } \chi \text{ and molecules } \gamma, \right. \\ &\quad \left((\chi, \text{tp}(\alpha, \gamma)) \in \varphi \Rightarrow \text{In}(\chi, \psi, \text{tp}(\gamma, \beta)) \right. \\ &\quad \left. \left. \text{and } (\chi, \text{tp}(\beta, \gamma)) \in \psi \Rightarrow \text{In}(\chi, \varphi, \text{tp}(\gamma, \alpha)) \right) \right] \end{aligned}$$

$$\begin{aligned} \text{In}(\varphi, \psi, \tau) &\iff \\ &\psi \text{ is a set and, for some form } \chi \text{ and molecule } \gamma, \\ &(\chi, \text{tp}(\beta, \gamma)) \in \psi \text{ and } \text{Eq}(\varphi, \chi, \text{tp}(\alpha, \gamma)) \end{aligned}$$

We prove these definitions correct simultaneously by induction on $\text{rank } \varphi + \text{rank } \psi$.

(Eq) We may assume that both φ and ψ are sets as the other two cases are trivial.

(\Rightarrow) Suppose $\varphi \star \alpha = \psi \star \beta$ and let χ be a form and γ a molecule. Suppose $(\chi, \text{tp}(\alpha, \gamma)) \in \varphi$. Then $\chi \star \gamma \in \varphi \star \alpha = \psi \star \beta$. Since $\text{rank } \chi < \text{rank } \varphi$, it follows from the inductive hypothesis that $\text{In}(\chi, \psi, \text{tp}(\gamma, \beta))$ holds. The case $(\chi, \text{tp}(\beta, \gamma)) \in \psi$ is symmetric.

(\Leftarrow) Suppose $\text{Eq}(\varphi, \psi, \text{tp}(\alpha, \beta))$ holds. We show that $\varphi \star \alpha \subseteq \psi \star \beta$ and the result follows by symmetry. Suppose $\chi \star \gamma \in \varphi \star \alpha$. We must have $(\chi, \text{tp}(\alpha, \gamma)) \in \varphi$ so, by definition of Eq, $\text{In}(\chi, \psi, \text{tp}(\gamma, \beta))$ holds. By the inductive hypothesis, $\chi \star \gamma \in \psi \star \beta$.

(In) The result is trivial if ψ is not a set so assume that it is. $\varphi \star \alpha \in \psi \star \beta$ if, and only if, there is a $(\chi, \tau) \in \psi$ with $\varphi \star \alpha = \chi \star \tau$ for some τ with $\text{tp}(\beta, \tau) = \tau$. By the inductive hypothesis, this happens if, and only if, $\text{Eq}(\varphi, \chi, \text{tp}(\alpha, \tau))$, if, and only if, $\text{In}(\varphi, \psi, \text{tp}(\alpha, \beta))$. \square

We are now ready to prove the equivalence theorem.

Proof of Theorem 33. We show that $\bar{I}_k \equiv^{\mathcal{C}^m} \bar{J}_k$ by exhibiting a winning strategy for the duplicator in the m -pebble bijective game on the two structures. The strategy is to ensure that, after every move, there are forms $\varphi_1, \dots, \varphi_m$, I -molecules $\alpha_1, \dots, \alpha_m$ and J -molecules β_1, \dots, β_m such that, for every $i \in \{1, \dots, m\}$, $x_i = \varphi_i \star \alpha_i$, $y_i = \varphi_i \star \beta_i$ and $\text{tp}_I(\alpha_1, \dots, \alpha_m) = \text{tp}_J(\beta_1, \dots, \beta_m)$, where the x_i and y_i are the pebbled elements of \bar{I}_k and \bar{J}_k , respectively.

We show first that the strategy can be followed and then that it is winning. We may assume that, in the initial position, all pebbles are on \emptyset . Put all the $\varphi_i = \psi_i = \emptyset$; by Lemma 34, we can set the α_i arbitrarily and choose the β_i with the same types. The condition holds.

Suppose the condition holds after some number of moves and that, without loss of generality, the spoiler chooses to play the pebbles x_1 and y_1 . We define the duplicator's bijection $f : \bar{I}_k \rightarrow \bar{J}_k$.

First, \mathcal{C}^{mk} -equivalence of I and J means that there is a bijection $g : |I|^k \rightarrow |J|^k$

such that, for all $\alpha \in |I|^k$, $\text{tp}_I(\alpha, \alpha_2, \dots, \alpha_m) = \text{tp}_J(g(\alpha), \beta_2, \dots, \beta_m)$. To see this, consider the position in the game on I and J , where the pebbles in I are on $\alpha_{21}, \dots, \alpha_{mk}$ and suppose the spoiler plays pebbles in k successive rounds to the elements of $\alpha = a_1, \dots, a_k$. The duplicator's winning strategy gives her bijections $h_1, h_2[a_1], h_3[a_1, a_2], \dots, h_k[a_1, \dots, a_{k-1}]$ where the bijections chosen after the first round depend on the spoiler's placement of the pebbles in the earlier rounds. $g(\alpha) = \langle h_1(a_1), h_2[a_1](a_2), \dots, h_k[a_1, \dots, a_{k-1}](a_k) \rangle$.

Second, by Lemma 39, for all forms φ and φ' and molecules α and α' , whether $\varphi \star \alpha = \varphi' \star \alpha'$ depends only on the two forms and $\text{tp}(\alpha, \alpha')$. Since $\text{tp}_I(\alpha, \alpha') = \text{tp}_J(g(\alpha), g(\alpha'))$, it follows that $\varphi \star \alpha = \varphi' \star \alpha'$ if, and only if, $\varphi \star g(\alpha) = \varphi' \star g(\alpha')$.

By Lemma 38, every $x \in \bar{I}_k$ can be written $\varphi_x \star \alpha_x$. We may choose values arbitrarily for each φ_x and α_x and set $f(\varphi_x \star \alpha_x) = \varphi_x \star g(\alpha_x)$. It remains to check that f is a bijection. f is one-to-one because of the relation Eq. To establish that f is onto, let $y \in \bar{J}_k$, choose ψ and β such that $y = \psi \star \beta$ and let $x = \psi \star g^{-1}(\beta) \in \bar{I}_k$. But now $f(x) = \varphi_x \star g(\alpha_x) = \psi \star \beta$, again by the relation Eq.

When the spoiler places the pebbles on elements $x \in \bar{I}_k$ and $f(x) = y \in \bar{J}_k$, the duplicator sets $\varphi_1 = \varphi_x$, $\alpha_1 = \alpha_x$, and $\beta_1 = g(\alpha_x)$.

It remains to prove that the duplicator's strategy actually wins the game, i.e., that the map $h : x_1 \dots x_m \mapsto y_1 \dots y_m$ is always a partial isomorphism. h preserves the structure of I because $\text{tp}_I(\alpha_1, \dots, \alpha_m) = \text{tp}_J(\beta_1, \dots, \beta_m)$; it just remains to show that the set-structure is preserved. By Lemma 39,

$$\begin{aligned} x_i = x_j &\iff \varphi_i \star \alpha_i = \varphi_j \star \alpha_j \\ &\iff \text{Eq}(\varphi_i, \varphi_j, \text{tp}_I(\alpha_i, \alpha_j)) \\ &\iff \text{Eq}(\varphi_i, \varphi_j, \text{tp}_J(\beta_i, \beta_j)) \\ &\iff \varphi_i \star \beta_i = \varphi_j \star \beta_j \\ &\iff y_i = y_j \end{aligned}$$

and similarly for \in and In. \square

Finally, we recall the main result of Cai et al. They show that, for any k , there are arbitrarily large 3-regular graphs G such that $\mathfrak{G}^0 \equiv^{C^k} \mathfrak{G}^1$ [6]. The relevance of the graphs being 3-regular is that, for any of the graphs, $\|\mathfrak{G}^0\| = \|\mathfrak{G}^1\| = 7\|G\|$. (For a graph G of minimum degree δ and maximum degree Δ , $(\delta + 2^{\delta-1})\|G\| \leq \|\mathfrak{G}^0\| = \|\mathfrak{G}^1\| \leq (\Delta + 2^{\Delta-1})\|G\|$.)

We are now ready to pull the various elements together to prove that no $\tilde{\text{CPT}}(\text{Card})$ program that only activates sets of small rank can compute the parity of CFI graphs.

Theorem 40 *The parity query for pre-ordered CFI graphs is not accepted by any program of $\tilde{\text{CPT}}(\text{Card})$ that activates sets of rank at most $o\left(\frac{\log n}{\log \log n}\right)$.*

PROOF. Suppose the $\tilde{\text{CPT}}(\text{Card})$ program $\bar{\Pi}$ accepts all structures \mathfrak{G}^0 but activates no set of rank greater than r . By Corollary 21, there is an m such that, if $\bar{\Pi}$ accepts a structure I and $\text{Active}^+(I) \equiv^{C^m} \text{Active}^+(J)$, then $\bar{\Pi}$ accepts J . By the result of Cai et al., for any large enough n , we may choose a 3-regular graph G with $\mathcal{O}(n)$ vertices such that $\mathfrak{G}^0 \equiv^{C^n} \mathfrak{G}^1$. Moreover, by Corollary 31, there is a c such that, if G is large enough and $x \in \text{Active}^+(\mathfrak{G}^0)$ or $\text{Active}^+(\mathfrak{G}^1)$, x has a support of size at most $c(\log_2 n)^r$.

By Theorem 33, $\text{Active}^+(\mathfrak{G}^0) \equiv^{C^m} \text{Active}^+(\mathfrak{G}^1)$ for all $m < \frac{n}{c(\log n)^r}$. If $r = o\left(\frac{\log n}{\log \log n}\right)$, then $\frac{n}{c(\log n)^r}$ is unbounded. Thus, if G is large enough, $\bar{\Pi}$ also accepts \mathfrak{G}^1 . \square

In particular, this implies that no $\tilde{\text{CPT}}(\text{Card})$ program using only sets of rank bounded by some constant can compute the CFI query. Since, by Theorem 22, any formula of $\text{IFP} + \text{C}$ can be translated into a $\tilde{\text{CPT}}(\text{Card})$ program that uses set of rank at most 1, the following corollary is a strengthening of the main result of Cai et al.

Corollary 41 *Let $r \in \mathbb{N}$. The parity query for pre-ordered CFI graphs is not defined by any $\tilde{\text{CPT}}(\text{Card})$ program that only activates sets of rank r or less.*

Clearly, these results continue to apply in the absence of the pre-order: any algorithm that can compute the parity of unordered CFI graphs can trivially compute the parity in the presence of a pre-order by ignoring the ordering. Further, the addition of a pre-order cannot increase the number of automorphisms of the input structure so cannot increase the resources required.

9 Concluding remarks

Our main results are that Blass, Gurevich and Shelah's language $\tilde{\text{CPT}}$ (without counting) can determine the parity of pre-ordered Cai–Fürer–Immerman graphs but that this cannot be done, even with counting, by any program that activates sets of rank bounded by some constant. In fact, for graphs G of order n , our program activates sets of rank $\mathcal{O}(n)$ to determine the parity of input \mathfrak{G}^T .

The algorithm crucially relies on the presence of the pre-order on the CFI graphs and it remains open whether there is a $\tilde{\text{CPT}}$ or $\tilde{\text{CPT}}(\text{Card})$ algorithm

that determines the parity of unordered CFI graphs or, indeed, whether there is any polynomial-time query not definable in $\tilde{\text{CPT}}(\text{Card})$. Our algorithm can clearly be adapted to work on any class of graphs where an order is definable and can also be modified to work for some other classes of unordered graphs, such as complete graphs, which are the examples presented by Blass et al. in [13].

Acknowledgements

The authors would like to thank Alan Nash for the helpful comments he made on a draft of this paper.

An extended abstract of this paper appeared as [20]. We are grateful to the referee for pointing out a serious error in Section 5 of that paper. This section has been completely re-written as Section 7 of the present paper.

References

- [1] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets, in: R. M. Karp (Ed.), *Complexity of Computation*, Vol. 7 of SIAM-AMS Proceedings, 1974, pp. 43–73.
- [2] N. Immerman, Relational queries computable in polynomial time, *Information and Control* 68 (1–3) (1986) 86–104.
- [3] M. Y. Vardi, The complexity of relational query languages, in: *Proc. 14th ACM Symp. on Theory of Computing*, 1982, pp. 137–146.
- [4] Y. Gurevich, S. Shelah, Fixed-point extensions of first-order logic, *Annals of Pure and Applied Logic* 32 (1986) 265–280.
- [5] S. Kreutzer, Expressive equivalence of least and inflationary fixed-point logic, in: *Proc. 17th IEEE Symp. on Logic in Computer Science (LICS)*, 2002, pp. 403–410.
- [6] J.-Y. Cai, M. Fürer, N. Immerman, An optimal lower bound on the number of variables for graph identification, *Combinatorica* 12 (4) (1992) 389–410.
- [7] H.-D. Ebbinghaus, J. Flum, *Finite Model Theory*, 2nd Edition, Springer, 1999.
- [8] N. Immerman, *Descriptive Complexity*, Springer, 1999.
- [9] F. Gire, H. Hoang, An extension of fixpoint logic with a symmetry-based choice construct, *Information and Computation* 144 (1) (1998) 40–65.

- [10] A. Dawar, D. M. Richerby, A fixed-point logic with symmetric choice, in: Proc. 17th Workshop on Computer Science Logic, Vol. 2803 of LNCS, Springer, 2003, pp. 169–182.
- [11] A. Dawar, D. M. Richerby, Fixed-point logics with nondeterministic choice, *Journal of Logic and Computation* 13 (4) (2003) 503–530.
- [12] A. Blass, Y. Gurevich, S. Shelah, Choiceless polynomial time, *Annals of Pure and Applied Logic* 100 (1–3) (1999) 141–187.
- [13] A. Blass, Y. Gurevich, S. Shelah, On polynomial time computation over unordered structures, *Journal of Symbolic Logic* 67 (3) (2002) 1093–1125.
- [14] M. Otto, *Bounded Variable Logics and Counting*, Vol. 9 of Lecture Notes in Logic, Springer, 1997.
- [15] P. G. Kolaitis, M. Y. Vardi, Infinitary logics and 0-1 laws, *Information and Computation* 98 (2) (1992) 258–294.
- [16] E. Grädel, M. Otto, Inductive definability with counting on finite structures, in: *Selected papers from 6th Workshop on Computer Science Logic*, Vol. 702 of LNCS, Springer, 1993, pp. 231–247.
- [17] L. Hella, Logical hierarchies in PTIME, *Information and Computation* 129 (1) (1996) 1–19.
- [18] Y. Gurevich, Evolving algebra 1993: Lipari guide, in: E. Börger (Ed.), *Specification and Validation Methods*, Oxford University Press, 1995, pp. 9–36.
- [19] J. Flum, M. Grohe, On fixed-point logic with counting, *Journal of Symbolic Logic* 65 (2) (2000) 777–787.
- [20] A. Dawar, D. M. Richerby, B. Rossman, Choiceless polynomial time, counting and the Cai–Fürer–Immerman graphs (extended abstract), in: *Proc. 12th Workshop on Logic, Language, Information and Computation*, Vol. 143 of ENTCS, Elsevier, 2005, pp. 13–26.