Choiceless Polynomial Time, Counting and the Cai–Fürer–Immerman Graphs (Extended Abstract)

Anuj Dawar,\textsuperscript{a,1} David Richerby\textsuperscript{a,2} and Benjamin Rossman\textsuperscript{3}

\textsuperscript{a} University of Cambridge Computer Laboratory, William Gates Building, J.J. Thomson Avenue, Cambridge, CB3 0FD, United Kingdom

Abstract

We consider Choiceless Polynomial Time (CPT), a language introduced by Blass, Gurevich and Shelah, and show that it can express a query originally constructed by Cai, F{"u}rer and Immerman to separate fixed-point logic with counting (IFP + C) from 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 CPT(Card), an extension of CPT with counting.

Keywords: Descriptive complexity, finite model theory, counting, choiceless polynomial time.

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. Fagin has shown that a class of structures is decidable in \textbf{NP} if, and only if, it is definable in existential second-order logic \cite{6}; the central open question is whether such a characterization of deterministic polynomial time exists.

\textsuperscript{1} E-mail: Anuj.Dawar@cl.cam.ac.uk
\textsuperscript{2} E-mail: David.Richerby@cl.cam.ac.uk. Supported by EPSRC grant GR/S06721.
\textsuperscript{3} E-mail: brossman@impa.br. Research carried out during an internship at Microsoft Research, One Microsoft Way, Redmond, WA 98052, U.S.A.
Immerman [10] and Vardi [12] independently showed that inflationary fixed-point logic (IFP) captures polynomial time on ordered structures; however, without a linear order, there is no IFP formula that expresses that the size of a structure is even. Cai, Fürer and Immerman showed that the extension IFP + C of IFP with counting terms is still too weak to capture \( P \) [3]. For any finite, connected graph \( G \), they define a pair of ‘CFI graphs’ \( \mathcal{G}^0 \) (known as ‘even’) and \( \mathcal{G}^1 \) (‘odd’) that are rich in automorphisms, distinguishable in polynomial time but not, for a suitable series of graphs \( G \), distinguishable in IFP + C. The construction can also be applied to ordered graphs, in which case \( \mathcal{G}^0 \) and \( \mathcal{G}^1 \) are pre-ordered. Proofs of these results and a fuller account of the problem of logically characterizing \( P \) can be found in, e.g., [5,11].

Gire and Hoang introduced an extension of IFP with a nondeterministic choice operator that is constrained to produce a deterministic logic [7] (see also [4]). This logic can distinguish the CFI graphs and it remains open whether it captures \( P \). In the other direction, Blass, Gurevich and Shelah introduce \emph{choiceless polynomial time} with the aim of characterizing what can be done without choice [1]. This logic is based on a machine model and is strictly more expressive than IFP. It still cannot define simple cardinality queries so the same authors also study an extension \( \tilde{\text{CPT}}(\text{Card}) \) with counting terms [2]. This is strictly more powerful than IFP + C: in particular, Blass et al. show that it can distinguish suitably padded versions of the CFI graphs. They leave open the question of whether \( \tilde{\text{CPT}}(\text{Card}) \) captures \( P \) and, in particular, whether the unpadded CFI graphs are distinguishable in the logic.

In this paper, we answer one of these questions by showing that \( \tilde{\text{CPT}} \) can distinguish the odd and even CFI graphs of ordered, connected graphs. Our algorithm constructs objects (hereditarily finite sets) that have a high degree of symmetry but are still able to determine the parity of the graph. The algorithm does not use counting but crucially relies on the use of sets of high rank. We give a corresponding lower bound by showing that no \( \tilde{\text{CPT}}(\text{Card}) \) program using only objects of bounded rank can define the parity query for the CFI graphs even of ordered graphs. This is based on an analysis of the automorphism groups of the CFI graphs, extending the techniques developed by Blass et al. to show that evenness is not \( \tilde{\text{CPT}} \)-definable.

We assume familiarity with standard extensions of first order logic with fixed-point operators and counting and the relationships between these extensions and infinitary logics; see, e.g., [5]. 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 \( J \).
2 The graphs

We now describe the class of graphs, originally due to Cai, Füredi and Immerman [3], that we use in the rest of the paper; we follow Blass et al. [2].

For a graph $G = (V, E)$, assumed to be finite, undirected and simple, $V(G) = V$, $v(G) = |V|$ and $E(G) = E$; we write $E(v)$ for the set of edges incident on $v$ and $\delta(G)$ for the minimal degree of $G$’s vertices.

**Definition 2.1** Let $G = (V, E)$ be a connected graph with at least two vertices. Let $\hat{V} = \{ v^X : v \in V$ and $X \subseteq E(v) \}$ and let $\hat{E} = \{ e^0, e^1 : e \in E \}$, where the $v^X$ and $e^i$ are new atoms (i.e., primitive entities considered not to be sets). $G^* = (\hat{V} \cup \hat{E}, \{ \{ v^X, e^1 \} : e \in X \} \cup \{ \{ v^X, e^0 \} : e \not\in X \})$.

For $v \in V$, write $v^*$ for the associated vertices in $V^*$; put $e^* = \{ e^0, e^1 \}$.

**Definition 2.2** 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 $\mathfrak{G} = (G^*, \leq)$, where the linear pre-order $\leq$ is defined by putting $v^X \leq w^Y$ for $v \leq w$, $e^i \leq f^j$ for $e \leq f$ and $v^X \leq e^i$ for all $v^X$ and $e^i$.

The action of any automorphism $\rho$ of $\mathfrak{G}$ is completely determined by the set $\{ e : \rho$ swaps $e^0$ and $e^1 \}$. Indeed, for every $H = (V', E') \subseteq G$ (not necessarily an induced subgraph), we can define $\rho_H$ to be the automorphism of $\mathfrak{G}$ that flips exactly those edges in $E'$: $\rho_H(e^i) = e^{1-i}$ for $e \in E'$, $\rho_H(v^X) = v^{X \Delta (E' \cap E(v))}$ and $\rho_H(x) = x$, otherwise. Each $\rho_H$ depends only on $E(H)$ and is an involution of $\mathfrak{G}$. $\text{Aut}(\mathfrak{G})$ is generated by the set of $\rho_H$ where $H$ contains a single edge.

The CFI graphs are subgraphs of $\mathfrak{G}$ which have restricted automorphisms.

**Definition 2.3** Let $T \subseteq V$. For each $v \in T$, let $v^T = \{ v^X : ||X||$ is odd $\}$ and, for each $v \in V \setminus T$, let $v^T = \{ v^X : ||X||$ is even $\}$. $\mathfrak{G}^T$ is the subgraph of $\mathfrak{G}$ induced by $\hat{E} \cup \bigcup_{v \in V \setminus T} v^T$. $\mathfrak{G}^T$ is even if $||T||$ is even and odd, otherwise.

Since the $\rho_H$ are automorphisms of $\mathfrak{G}$, the image of any $\mathfrak{G}^T \subseteq \mathfrak{G}$ under $\rho_H$ must be 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. $||\text{odd}(H)||$ must be even; call $H$ even if $\text{odd}(H) = \emptyset$. In fact, for every $H \subseteq G$, $\rho_H(\mathfrak{G}^T) = \mathfrak{G}^{T \Delta \text{odd}(H)}$ and, for every $T \subseteq V$, $\text{Aut}(\mathfrak{G}^T) = \{ \rho_H : H \subseteq G$ is even $\}$. For connected $G$, $\mathfrak{G}^S \cong \mathfrak{G}^T$ if, and only if, $||S|| \equiv ||T||$ $(\text{mod } 2)$ (see [2]) so the two versions of $\mathfrak{G}^T$ are uniquely defined up to isomorphism: call these $\mathfrak{G}^0$ and $\mathfrak{G}^1$, respectively.

**Lemma 2.4** The automorphism group of $\mathfrak{G}^0$ (which is the same as $\text{Aut}(\mathfrak{G}^1)$) is generated by $\{ \rho_C : C \subseteq G$ is a cycle $\}$. 
3. \( \tilde{\text{CPT}}(\text{Card}) \)

The \( \tilde{\text{CPT}} \) model of computation is introduced by Blass, Gurevich and Shelah in [1] and extended with a counting mechanism by the same authors in [2]. We summarize the computation model here but the reader should consult these two references, particularly [1], for a full description.

Given an input structure \( I \) of vocabulary \( \sigma \), a \( \tilde{\text{CPT}} \) program operates over \( \text{HF}(I) \), the set of 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. Note that \( \text{HF}(I) \) contains the natural numbers, coded as von Neumann ordinals.

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 1. At this point, the program is deemed to accept if, and only if, the distinguished nullary dynamic function Output is 1.

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

3.1 States

A computation of a program with variables \( v_1, \ldots, v_k \) over input structure \( I \) is a finite or countable sequence of states \( S_0, S_1, \ldots \), where each state is a structure of vocabulary \( (\delta, \in, \emptyset, v_1, \ldots, v_k) \) with universe \( \text{HF}(I) \), with the binary relation \( \in \) and constant \( \emptyset \) are interpreted in the obvious way. Each \( v_i \) is a constant symbol whose interpretation is the value of the corresponding variable in the state. The initial state, \( S_0 \) interprets every dynamic function as the constant zero function.

3.2 Programs and runs

The terms over vocabulary \( (\sigma, \delta) \) are defined as follows. 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 the numbers 0 and 1, respectively.

**Boolean combinations.** If \( t_1 \) and \( t_2 \) are terms, then \( \lnot t_1 \) and \( t_1 \land t_2 \) are terms.

The Boolean connectives have the obvious denotation if their arguments take values 0 or 1 and denote 0, otherwise.

**Equality.** If \( t_1 \) and \( t_2 \) are terms, then \( t_1 = t_2 \) is a term.
Set-theoretic functions. $\emptyset$ and Atoms are terms; if $t_1$ and $t_2$ are terms, then $\bigcup t_1$, TheUnique($t_1$), $t_1 \in t_2$ and $\{t_1, t_2\}$ are terms. Atoms denotes the set of atoms 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 $[t]^S$ as a von Neumann ordinal, or 0 if $t$ denotes an atom.

Predicates. If $R \in \sigma$ is an $n$-ary relation symbol and $t_1, \ldots, t_n$ are terms, then $R(t_1, \ldots, t_n)$ is a Boolean-valued term.

Dynamic functions. If $f \in \delta$ is an $n$-ary dynamic function and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ 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 bound. $\[T\] = \{[t]^S_{a/v} : a \in [r]^S$ and $[g]^S_{a/v} = \text{true}\}$.

A $\tilde{\text{CPT}}(\text{Card})$ program is a rule without free variables; a $\tilde{\text{CPT}}(\text{Card})$ program containing no Card terms. We define the semantics of rules only informally. The rule Skip does nothing. For an $n$-ary dynamic function $f$, the rule $f(t_1, \ldots, t_n) := t_{n+1}$ sets the value of $f([t_1], \ldots, [t_n])$ to $[t_{n+1}]$. For a Boolean-valued term $t$, if $t$ then $R_1$ else $R_2$ fi is a rule and, for a variable $v$ not occurring in term $t$, do forall $v \in t R$ od is a rule (in which $v$ is bound) that executes $R(a)$ in parallel for each $a \in [t]$. The new state is determined by simultaneously performing all updates determined by the rules, except that, if two updates contradict each other, no updates are performed so the state is unchanged and the run does not terminate. If Halt is set to 1, execution terminates.

3.3 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 HF($I$) it is allowed to use. Both bounds are necessary to ensure that $\tilde{\text{CPT}}(\text{Card}) \subseteq \text{P}$.

The active objects in a state are 0, 1, elements of the ranges of the dynamic functions, elements of tuples mapped by any dynamic function to a non-zero value and any element of the transitive closure of an otherwise active element. Write Active($I$) for the substructure of HF($I$) containing all the objects that become active when $\Pi$ is run on $I$. Because of the choiceless nature of the computation, Active($I$) is closed under all automorphisms of $I$.

Definition 3.1 A $\tilde{\text{CPT}}(\text{Card})$ program of input vocabulary $\sigma$ is a tuple $\tilde{\Pi} = (\Pi, p, q)$, where $\Pi$ is a program and $p, q \in \mathbb{N}$, such that, for any input $I$ of
vocabulary $\sigma$, the run of $\Pi$ on $I$ has at most $\|I\|^p$ steps and $\|\text{Active}(I)\| \leq \|I\|^q$.

### 3.4 Fixed-point definability

**Theorem 3.2** Let $\overline{\Pi} = (\Pi, p, q)$ be a program with input $I$. There is a formula $\varphi \in \text{IFP} + C$ such that $HF(I) \models \varphi$ if, and only if, $\overline{\Pi}$ accepts $I$.

The proof is a relatively straightforward adaptation of the corresponding proof for $\tilde{\text{CPT}}$ and IFP in [1]. In fact, $\varphi$ does not need access to the whole of $HF(I)$ but only needs the elements of $\text{Active}(I)$ with the von Neumann ordinals up to $\|I\|^p$ added to the set structure, to number the stages. Call this structure $\text{Active}^+(I)$.

**Corollary 3.3** Let $\overline{\Pi} = (\Pi, p, q)$ be a program with input $I$. There is a formula $\varphi \in \text{IFP} + C$ such that $\text{Active}^+(I) \models \varphi$ if, and only if, $\overline{\Pi}$ accepts $I$.

### 4 The algorithm

We now present a $\tilde{\text{CPT}}$ algorithm that determines the parity of graphs $\mathcal{G}^T$ (that is, determines the parity of $\|T\|$). The algorithm does not require counting but does use a slightly enriched model of computation. A $\tilde{\text{CPT}}$ program with input structure $I$ ordinarily runs on $HF(I)$, 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 4.1** Let $I$ be a set. $HF^+(I)$ is the least set containing every element of $|I|$ along with 0 and 1 as atoms and closed under the operations of forming finite subsets and tuples of finite length.

In other words, $HF^+(I)$ treats tuples and the numbers 0 and 1 as first-class objects, rather than coding them as sets. However, because these new objects can be efficiently coded as sets, using this enriched universe does not affect the expressive power of programs. The notion of rank extends to $HF^+(I)$ in the obvious way: atoms have rank zero; sets and tuples have rank one higher than the greatest rank of their elements.

For the remainder of this section, fix a finite, connected, ordered graph $G = (V, E, \leq)$ and let $v_1, \ldots, v_n$ enumerate $V$ according to the linear order $\leq$. Recall that automorphisms of the graph $\mathcal{G}$ are given by $\rho_H$ where $H \subseteq G$ and automorphisms of a Cai–F¨urer–Immerman graph $\mathcal{G}^T$ are precisely those $\rho_H$ where $H \subseteq G$ is even. Each $\rho_H \in \text{Aut}(\mathcal{G})$ naturally induces a bijection $HF^+(\widehat{E}) \rightarrow HF^+(\widehat{E})$ (that extends the restriction of $\rho_H$ to $\widehat{E}$).
Definition 4.2 \( x \in \text{HF}^+(\hat{E}) \) 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 \( v^X \) is adjacent in \( \mathfrak{G} \) to exactly one of the vertices \( e^0 \) and \( e^1 \) for each edge \( e \) adjacent to \( v \) in \( G \). Let \( N(u) \) be the set of neighbours of \( u \) in \( \mathfrak{G} \), and let \( N_\prec(u) \) 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.) Let \( \tilde{N}(u) = \{ e^{1-i} : e^i \in N(u) \} \) and define \( \tilde{N}_\prec(u) \) similarly to \( N_\prec(u) \).

Definition 4.3 For all \( T \subseteq V \) and \( i \in \{1, \ldots, n\} \), let
\[
\tau^T_i = \{ N_\prec(u) : u \in v^T_i \}
\]
\[
\tilde{\tau}^T_i = \{ \tilde{N}_\prec(u) : u \in v^T_i \}.
\]
Let \( \mu^T_1 = \tau^T_1 \) and \( \tilde{\mu}^T_1 = \tilde{\tau}^T_1 \). For \( i \in \{1, \ldots, n-1\} \), let
\[
\mu^T_{i+1} = \{ \langle \mu^T_i, \tau^T_{i+1} \rangle, \langle \tilde{\mu}^T_i, \tilde{\tau}^T_{i+1} \rangle \}
\]
\[
\tilde{\mu}^T_{i+1} = \{ \langle \mu^T_i, \tilde{\tau}^T_{i+1} \rangle, \langle \tilde{\mu}^T_i, \tau^T_{i+1} \rangle \}.
\]

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

Lemma 4.4 There is a CPT program which, given input structure \( \mathfrak{G}^T \), outputs the object \( \mu^T_n \) in \( \|E\| + n \) steps, activating \( O(\|\mathfrak{G}^T\|) \) objects.

Proof. Construct unary dynamic functions \( N_\prec \) and \( \tilde{N}_\prec \) over \( \{ v^T : v \in V \} \), in \( O(\|E\|) \) steps, activating \( O(\|\mathfrak{G}^T\|) \) objects. The maps \( v^X_i \mapsto \tau^T_i \) and \( v^X_i \mapsto \tilde{\tau}^T_{i+1} \) are then given by suitable comprehension terms. Now construct the maps \( v^X_i \mapsto \mu^T_i \) and \( v^X_i \mapsto \tilde{\mu}^T_i \) in turn in \( n \) more steps and output \( \mu^T_n \).

Lemma 4.5 For all \( S, T \subseteq V \) and \( k \in \{1, \ldots, n\} \),
\[
\begin{align*}
(i) \quad & \tau^S_k = \tau^T_k \iff \tilde{\tau}^S_k = \tilde{\tau}^T_k \iff v_k \notin S \triangle T; \\
(ii) \quad & \tau^S_k = \tilde{\tau}^T_k \iff \tilde{\tau}^S_k = \tau^T_k \iff v_k \in S \triangle T.
\end{align*}
\]

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

Lemma 4.6 For all \( S, T \subseteq V \) and \( k \in \{1, \ldots, n\} \),
\[
\begin{align*}
(i) \quad & \mu^S_k = \mu^T_k \iff \tilde{\mu}^S_k = \tilde{\mu}^T_k \iff \|S(k)\| \equiv \|T(k)\| \quad (\text{mod 2}); \\
(ii) \quad & \mu^S_k = \tilde{\mu}^T_k \iff \tilde{\mu}^S_k = \mu^T_k \iff \|S(k)\| \neq \|T(k)\| \quad (\text{mod 2}).
\end{align*}
\]

Corollary 4.7 For all \( T \subseteq V \), \( \mu^T_n \) is super-symmetric.

Proof. Any \( \rho_H \in \text{Aut}(\mathfrak{G}) \) maps \( \mathfrak{G}^T \) to \( \mathfrak{G}^S \), where \( S = T \triangle \text{odd}(H) \) and, therefore, maps \( \mu^T_n \) to \( \mu^S_n \). Since \( \|T\| \equiv \|S\| \quad (\text{mod 2}) \), we have \( \mu^T_n = \mu^S_n \).
For each $e^i \in \hat{E}$, let $B_{e^i}$ be the function that maps an object $x$ to the object that results from recursively replacing every instance of $e^i$ in $x$ with 0 and every instance of $e^{1-i}$ with 1. Let $B(x) = B_{e^0}(B_{e^2}(\cdots B_{e^m}(x)\cdots))$, where $e_1, \ldots, e_m$ is the enumeration of $E$ in the order induced by $\leq$. We would like to compute $B(\mu^T_n)$ by a CPT algorithm. This calculation looks problematic, since $e^0_i$ and $e^1_i$ are indistinguishable in $\mathcal{G}^T$ up to isomorphism but we can compute $B(\mu^T_n)$ without having to isolate $e^0_1, \ldots, e^0_m$ from the rest of $\hat{E}$.

**Lemma 4.8** If $x \in \text{HF}^+(\hat{E})$ is super-symmetric then, for every $e \in E$, $B_{e^0}(x) = B_{e^1}(x)$ and is super-symmetric.

**Proof.** Let $x \in \text{HF}^+(\hat{E})$ be super-symmetric and let $e \in E$. We argue by induction on the rank of $x$. If $x = \langle y_1, \ldots, y_k \rangle$ then each $y_i$ is super-symmetric and we are done. For a set $x$, since $\rho_e(x) = x$, we have $x = x \cup \rho_e(x)$.

$$B_{e^0}(x) = B_{e^0}(x \cup \rho_e(x)) = B_{e^0}(x) \cup B_{e^0}(\rho_e(x)) = B_{e^1}(x) \cup B_{e^1}(\rho_e(x) \cup x) = B_{e^1}(x).$$

To prove super-symmetry, it suffices to show that $\rho_f$ fixes $B_{e^i}(x)$ for all $f \in E$. 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 $\rho_f(B_{e^i}(x)) = B_{e^i}(\rho_f(x)) = B_{e^i}(x)$. □

**Lemma 4.9** There is a $\widetilde{\text{CPT}}$ program that, for input $\mathcal{G}^T$, outputs $B(\mu^T_n)$.

**Proof.** First, compute $\mu^T_n$ using the $\text{CPT}$ program described in Lemma 4.4. We can then define a $\widetilde{\text{CPT}}$ program which, from input $x \in \text{HF}^+(\hat{E})$ and a distinguished atom $e^i$, computes $B_{e^i}(x)$ in $O(\text{rank}(x))$ steps, using $O(||TC(x)||)$ active objects, where $\text{TC}(x)$ is the transitive closure of $x$. It is, therefore, possible to compute the sequence $b_0, \ldots, b_m$ in $O(MS)$ additional steps, where $b_0 = \mu^T_n$ and $b_{i+1} = \text{TheUnique}(\{ B_{e^0_{i+1}}(b_i), B_{e^1_{i+1}}(b_i) \})$. Finally, output $b_m$: by Lemma 4.8, $b_m = B(\mu^T_n)$. □

Notice that $B(\mu^T_n) \in \text{HF}^+(\emptyset)$. We now define the function $p : \text{HF}^+(\emptyset) \to \{0, 1\}$ recursively by putting $p(0) = 0$, $p(1) = 1$, $p(\{x_1 \ldots x_k\}) = \prod_i p(x_i)$ and $p(\langle x_1 \ldots x_k \rangle) = \sum_i p(x_i) \pmod{2}$. Note that the arithmetic required to compute $p(x)$ can be performed in $\text{CPT}$, 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.

**Lemma 4.10** $p(B(\mu^T_n)) \equiv ||T|| \pmod{2}$.

**Proof.** By Lemma 4.6, it suffices to check the cases $T = \emptyset$ and $T = \{v_n\}$. 

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

Let $P(x) = p(B(x))$. We show by induction that, for all $i \in \{1, \ldots, n\}$, $P(\mu_i^0) = 0$ and $P(\mu_i^\emptyset) = 1$. The case $i = 1$ has already been dealt with, since $\mu_1 = \tau_1$ and $\tilde{\mu}_1 = \tilde{\tau}_1$. Suppose $P(\mu_i^0) = 0$ and $P(\mu_i^\emptyset) = 1$: it is easy to check that $P(\mu_i^{0+1}) = 0$ and $P(\mu_i^{\emptyset+1}) = 1$.

Similarly, for $T = \{v_n\}$, $P(\tau_n^T) = P(\mu_n^\emptyset) = 1$ and $P(\tau_n^T) = P(\tau_n^T) = 0$ and, for $1 \leq i < n$, $P(\tau_i^T) = P(\mu_i^T) = 0$ and $P(\tau_i^T) = P(\mu_i^T) = 1$. \hfill \Box

**Theorem 4.11** There is a ČPT algorithm that, given input structure $\mathfrak{S}^T$, outputs $\|T\|$ (mod 2).

**Proof.** Invoking Lemma 4.9, we first constructs $B(\mu_n^T)$. It is easy to see that, for $x \in HF^+(\emptyset)$, $p(x)$ can be computed in $\mathcal{O}(\text{rank}(x))$ steps activating $\mathcal{O}(\|TC(x)\|)$ objects. This allows us to compute and output $p(B(\mu_n^T))$. \hfill \Box

In the remainder of the paper, we revert to considering ČPT programs with input $I$ as working on $HF(I)$ rather than $HF^+(I)$.

## 5 Supports

Any automorphism $\rho$ of a structure $I$ can be extended inductively to an automorphism $\tilde{\rho}$ of $HF(I)$ by putting $\tilde{\rho}(x) = \{ \tilde{\rho}(y) : y \in x \}$; indeed, every automorphism of $HF(I)$ can be obtained in this way. From this point, we will not distinguish between $\rho$ and $\tilde{\rho}$. For any $x \in HF(I)$, let $\text{Orbit}(x)$ denote the set of images of $x$ under automorphisms of $I$.

**Definition 5.1** ([1]) A set $S$ of atoms is a **support** for an object $x \in HF(I)$ if $x$ is fixed by every automorphism of $I$ that pointwise fixes $S$.

A support need not fix $x$ pointwise. For example, the set of all atoms has empty support, as does any von Neumann ordinal.

For the remainder of this section, let $\Pi$ be a fixed ČPT(Card) program which activates at most $n^d$ objects for any input $I$ of size $n$. $\text{Active}^+(I)$ is closed under all automorphisms of $I$, so, if $x \in \text{Active}^+(I)$ then $\|\text{Orbit}(x)\| \leq n^d$ and, moreover, $\|\text{Orbit}(y)\| \leq n^d$ for any $y$ in the transitive closure of $x$, since $\text{Active}^+(I)$ is transitively closed.

In this section and the next, we will concentrate on the family of ordered, rectangular $n \times n$ toroidal grid graphs $G_n$. That is, $G_n$ consists of the set of nodes $\{(i, j) : 0 \leq i, j < n\}$ with all edges of the form $\{(i, j), (i + 1, j)\}$ and $\{((i, j), (i, j + 1))\}$ where addition is taken modulo $n$. Our aim now is to
prove that there exists a constant $c$ such that if $\Pi$ activates an object $x$ of rank $r$ over input structure $\mathcal{G}_n^0$ or $\mathcal{G}_n^1$, then $x$ has a support of size $O(c(\log n)^r)$. We establish this by an analysis of $\text{Aut}(\mathcal{G}_n^0) (= \text{Aut}(\mathcal{G}_n^1))$.

**Lemma 5.2** $\text{Aut}(\mathcal{G}_n^0)$ is generated by $\{ \rho_C : C \in S_n \}$, where $S_n$ is the set of 4-cycle subgraphs of $G_n$.

Note also that, if $S \subseteq S_n$ is a set of mutually edge-disjoint 4-cycles, the subgroup generated by $\{ \rho_C : C \in S \}$ has order $2^{|S|}$.

**Definition 5.3** For any $x \in \text{HF}(\mathcal{G}_n^0)$, pre-support $(x)$, the pre-support of $x$, is the set $\{ C \in S_n : \rho_C(x) \neq x \}$.

Consider the recursively-defined function $s(0) = 4$, $s(r + 1) = d(2s(r) + 9) \log n$. Clearly, there is a constant $c$ such that $s(r)$ is $O(c^r(\log n)^r)$, i.e., for fixed $r$, $s(r)$ is $O((\log n)^r)$.

**Lemma 5.4** If $x \in \text{Active}^+(\mathcal{G}_n^i)$ and rank $x \leq r$, then $||\text{pre-support}(x)|| \leq s(r)$.

**Proof (sketch).** By induction on $r$. If $r = 0$, $x$ is an atom or the empty set. At most four 4-cycles move $x$ so $||\text{pre-support}(x)|| \leq 4 = s(0)$.

Suppose that rank $x = r + 1$. Let pre-support $(x) = \{ C_1, \ldots, C_t \}$ and suppose, towards a contradiction, that $t > s(r + 1)$. For each $i \in \{1, \ldots, t\}$, choose $y_i \in x$ with $\rho_{C_i}(y_i) \not\in x$ (such elements must exist as $\rho_{C_i}(x) \neq x$). For each $y_i$, $||\text{pre-support}(y_i) \cap \text{pre-support}(x)|| \leq s(r)$ and it can be shown that there is a set $T \subseteq \{1, \ldots, t\}$ of size greater than $d \log n$ such that, for any $i \neq j \in T$, $C_i$ and $C_j$ share no edges and $C_i \not\subseteq \text{pre-support}(y_j)$. The subgroup of automorphisms generated by $\{ \rho_{C_i} : i \in T \}$ has more than $n^d$ elements, no two of which map $x$ to the same point. This means that $||\text{Orbit}(x)|| > n^d$, contradicting the assumption that $x \in \text{Active}^+(\mathcal{G}_n^i)$. \hfill $\square$

This bound on the size of pre-supports for active objects of a given rank allows us to bound the size of their supports.

**Theorem 5.5** For all $r$, there is a constant $c$ such that every $x \in \text{Active}^+(\mathcal{G}_n^i)$ with rank $x \leq r$ has a support of size at most $c(\log n)^r$.

**Proof.** Fix $c_0$ such that $s(r) < c_0(\log n)^r$ for large enough $n$. By Lemma 5.4, $||\text{pre-support}(x)|| \leq c_0(\log n)^r$. Let $S$ be the set of atoms of the form $e^0$ or $e^1$ in $\mathcal{G}_n^i$ such that the edge $e$ of $G_n$ belongs to a 4-cycle in pre-support $(x)$. $||S|| \leq 8c_0(\log n)^r$ and it is easy to check that $S$ supports $x$. \hfill $\square$
6 Equivalence

Given a \( \sigma \)-structure \( I \) and a constant \( k \), the transitivity \( k \)-supported elements of \( \text{HF}(I) \) are those \( x \) every element of whose transitive closure has a support of size at most \( k \).\(^4\) We write \( \bar{I}_k \) for both the transitivity \( k \)-supported part of \( \text{HF}(I) \) and the corresponding structure of vocabulary \( \langle \sigma, \in, \emptyset \rangle \).

We generalize Theorem 33 of [1] in two ways: we consider the counting logics \( C^m \) and \( C^{mk} \) and relax the hypothesis from requiring \( I \) and \( J \) to be pure sets, allowing any pair of \( C^{mk} \)-homogeneous structures, i.e., structures \( I \) in which, whenever \( \bar{a}, \bar{b} \) in \( I \) have the same \( C^{mk} \)-type, there is an automorphism mapping \( \bar{a} \) to \( \bar{b} \). (Recall that the \( C^{mk} \)-type of a tuple \( \bar{a} \) in \( I \) is the collection of \( C^{mk} \) formulae satisfied by \( (I, \bar{a}) \).) In particular, \( \mathfrak{S}_n^0 \) and \( \mathfrak{S}_n^1 \) are \( C^n \)-homogeneous and it follows from results in [3] that, for any \( m \) and \( k \) and for sufficiently large \( n \), \( \mathfrak{S}_n^0 \equiv_c^{cm} \mathfrak{S}_n^1 \).

**Theorem 6.1** Let \( k, m \in \mathbb{N} \) and let \( I \) and \( J \) be \( C^{mk} \)-homogeneous structures of the same vocabulary. If \( I \equiv_c^{cm} J \) then \( \bar{I}_k \equiv_c^{cm} \bar{J}_k \).

Towards a proof, fix \( k, m, I \) and \( J \). An \( I \)-molecule is a sequence \( \alpha = \alpha_1 \ldots \alpha_k \) of not-necessarily distinct atoms. We will use molecules as supports. The \( n \)-ary type \( tp_I(\bar{\alpha}) \) of a sequence \( \bar{\alpha} = \alpha_1 \ldots \alpha_n \) of molecules is just the \( C^{nk} \)-type of the sequence of atoms \( \alpha_{11} \ldots \alpha_{mk} \) in \( I \). The following lemma is immediate from the definition of types and the fact that \( C^{mk} \)-types are determined by a single formula on finite structures [8].

**Lemma 6.2** For some \( \ell < m \), suppose that \( \alpha_0 \ldots \alpha_\ell \) are \( I \)-molecules and that \( \beta_1, \ldots, \beta_\ell \) are \( J \)-molecules. If \( tp_J(\beta_1, \ldots, \beta_\ell) = tp_I(\alpha_1, \ldots, \alpha_\ell) \), there is a \( J \)-molecule \( \beta_0 \) such that \( tp_J(\beta_0, \ldots, \beta_\ell) = tp_I(\alpha_0, \ldots, \alpha_\ell) \).

The definition of forms is adapted from that in [1]. Forms can be thought of as templates for building transitively \( k \)-supported sets from molecules.

**Definition 6.3** Fix a list \( c_1 \ldots c_k \) of new symbols. The set of forms is the least set containing each of the \( c_i \) and every finite set of pairs \( (\varphi, \tau) \), where \( \varphi \) is a form and \( \tau \) a binary type realized in \( I \).

The rank of a form is defined inductively: \( \text{rank } c_i = 0 \) and if \( \varphi \) is a set, \( \text{rank } \varphi = 1 + \max \{ \text{rank } \psi : (\psi, \tau) \in \varphi \} \). The denotation of a form \( \varphi \) and a molecule \( \alpha \) over a structure \( I \) is \( \varphi \star \alpha \in \text{HF}(I) \), where \( c_i \star \alpha = \alpha_i \) and \( \varphi \star \alpha = \{ \psi \star \beta : (\psi, tp_I(\alpha, \beta)) \in \varphi \} \) if \( \varphi \) is a set.

\(^4\) Blass et al. use the term ‘\( k \)-symmetric’ for the transitively \( k \)-supported elements; we avoid confusion with the ‘symmetric’ and ‘super-symmetric’ elements of Section 4.
The reason for using forms and molecules is that the objects denoted are exactly those in $I_k$. The proof of this result is similar to that of Lemma 39 in [1], though is simplified by the use of types rather than configurations.

**Lemma 6.4** $x \in \tilde{I}_k$ if, and only if, there is an $I$-molecule $\alpha$ and a form $\varphi$ such that $x = \varphi \star \alpha$.

The final fact we need is that there are relations $\mathsf{Eq}$ and $\mathsf{In}$ that allow us to determine whether $\varphi \star \alpha = (\text{resp., } \in) \psi \star \beta$ by considering only the forms $\varphi$ and $\psi$ and the types of $\alpha$ and $\beta$, independent of the structure from which the molecules come. The proof that these relations exist proceeds similarly to that of Lemma 40 in [1]. We are now ready to prove the equivalence theorem.

**Proof of Theorem 6.1 (sketch).** We outline a winning strategy for the duplicator in the $m$-pebble bijective game on $\tilde{I}_k$ and $\tilde{J}_k$, showing that the two structures are $C^{mk}$-equivalent [9]. We ensure that, after every move, there are forms $\varphi_1, \ldots, \varphi_m$, $I$-molecules $\alpha_1, \ldots, \alpha_m$ and $J$-molecules $\beta_1, \ldots, \beta_m$ such that, for each $i$, $x_i = \varphi_i \star \alpha_i$, $y_i = \varphi_i \star \beta_i$ and $\mathsf{tp}_I(\alpha_1, \ldots, \alpha_m) = \mathsf{tp}_J(\beta_1, \ldots, \beta_m)$.

Initially, all pebbles are on $\emptyset$ and the condition is easily satisfied. Suppose it holds for some position that has been reached and the spoiler chooses pebbles $x_1$ and $y_1$: we must define the duplicator’s bijection $h$. For each automorphism class of $\tilde{I}_k$ (considering only automorphisms that fix every $\alpha_i$ with $\varphi_i \neq \emptyset$), choose a canonical element $[x]$, which we can write $\varphi_{[x]} \star \alpha_{[x]}$ by Lemma 6.4. Each element of $[x]$’s automorphism class can be written $\varphi_{[x]} \star \rho_x(\alpha_{[x]})$ for some automorphism $\rho_x$. Lemma 6.2 gives the corresponding $\beta_{[x]}$ and because $I \equiv C^{mk} J$ and both are $C^{mk}$-homogeneous, $\beta_{[x]}$ has as many automorphic images in $J$ as $\alpha_{[x]}$ does in $I$. Pair them arbitrarily to give $h$ and use $\mathsf{In}$ and $\mathsf{Eq}$ to check that the map $x \mapsto y$ is a partial isomorphism so the strategy is winning. □

**Theorem 6.5** The Boolean query $\{ \mathcal{G}_n^0 : n \in \mathbb{N} \}$ is not accepted by any program of $\tilde{\mathsf{CPT}}(\mathsf{Card})$ that activates sets of rank at most $o(\frac{\log n}{\log \log n})$.

**Proof.** Suppose the $\tilde{\mathsf{CPT}}(\mathsf{Card})$ program $\tilde{\Pi}$ accepts all the structures $\mathcal{G}_n^0$ but activates no set of rank greater than $r$. By Corollary 3.3, there is an $m$ such that, if $\tilde{\Pi}$ accepts a structure $I$ and $\mathsf{Active}^+(I) \equiv C^m \mathsf{Active}^+(J)$, then $\tilde{\Pi}$ accepts $J$. Moreover, by Theorem 5.5, there is a $c$ such that, if $n$ is large enough and $x \in \mathsf{Active}^+(\mathcal{G}_n^0)$ or $\mathsf{Active}^+(\mathcal{G}_n^1)$, $x$ has a support of size at most $c(\log n)^r$. Since $\mathcal{G}_n^0 \equiv C^m \mathcal{G}_n^1$, by Theorem 6.1, $\mathsf{Active}^+(\mathcal{G}_n^0) \equiv C^m \mathsf{Active}^+(\mathcal{G}_n^1)$ for all $m < \frac{n}{c(\log n)^r}$. If $r = o(\frac{\log n}{\log \log n})$, then $\frac{n}{c(\log n)^r}$ is unbounded. Thus, $\tilde{\Pi}$ accepts $\mathcal{G}_n^1$ for all sufficiently large $n$. □

In particular, this implies that no $\tilde{\mathsf{CPT}}(\mathsf{Card})$ program using only sets of rank bounded by some constant can compute the CFI query. Since any
IFP + C formula naturally translates into a \( \tilde{\text{CPT}}(\text{Card}) \) program where the rank of sets used is bounded by a number that only depends on the formula, the following corollary is a strengthening of the main result of Cai et al.

**Corollary 6.6** Let \( r \in \mathbb{N} \). The Boolean query \( \{ \mathcal{G}_n^0 : n \in \mathbb{N} \} \) is not defined by any \( \tilde{\text{CPT}}(\text{Card}) \) program that only activates sets of rank \( r \) or less.

**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 CFI 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 \( O(n) \) to determine the parity of input \( \mathcal{G}_T \).

The algorithm crucially relies on the presence of the pre-order on the CFI graphs and it remains open whether there is a \( \text{CPT} \) or \( \tilde{\text{CPT}}(\text{Card}) \) algorithm that determines the parity of unordered CFI graphs. Our algorithm can clearly be adapted to work on any class of graphs where an order is definable, such as the toroidal grid graphs considered in Sections 5 and 6. A linear order can be defined on \( G_n \) given parameters interpreted as \((i, j), (i+1, j)\) and \((i, j+1)\) for any \( i, j \) — the graph with these parameters fixed has no non-trivial automorphisms. Our algorithm can also be modified to work for some other classes of graphs, such as complete graphs, the examples used in [2].

**References**


