Dynamic Complexity of Parity Exists Queries

Nils Vortmeier
TU Dortmund University, Dortmund, Germany
nils.vortmeier@tu-dortmund.de

Thomas Zeume
TU Dortmund University, Dortmund, Germany
thomas.zeume@tu-dortmund.de

Abstract

Given a graph whose nodes may be coloured red, the parity of the number of red nodes can easily be maintained with first-order update rules in the dynamic complexity framework DynFO of Patnaik and Immerman. Can this be generalised to other or even all queries that are definable in first-order logic extended by parity quantifiers? We consider the query that asks whether the number of nodes that have an edge to a red node is odd. Already this simple query of quantifier structure parity-exists is a major roadblock for dynamically capturing extensions of first-order logic.

We show that this query cannot be maintained with quantifier-free first-order update rules, and that variants induce a hierarchy for such update rules with respect to the arity of the maintained auxiliary relations. Towards maintaining the query with full first-order update rules, it is shown that degree-restricted variants can be maintained.

2012 ACM Subject Classification Theory of computation → Logic and databases; Theory of computation → Complexity theory and logic

Keywords and phrases Dynamic complexity, parity quantifier, arity hierarchy

Digital Object Identifier 10.4230/LIPIcs.CSL.2020.37

Related Version A full version of this paper is available at `http://arxiv.org/abs/1910.06004`.

Funding The authors acknowledge the financial support by DFG grant SCHW 678/6-2.

Acknowledgements We are grateful to Samir Datta, Raghav Kulkarni, Anish Mukherjee and Thomas Schwentick for illuminating discussions.

1 Introduction

The query Parity – given a unary relation U, does U contain an odd number of elements? – cannot be *expressed* in first-order logic, even with arbitrary numerical built-in relations [2, 9]. However, it can easily be *maintained* in a dynamic scenario where single elements can be inserted into and removed from U, and helpful information for answering the query is stored and updated by first-order definable update rules upon changes. Whenever a new element is inserted into or an existing element is removed from U, then a stored bit P is flipped[1]. In the dynamic complexity framework by Patnaik and Immerman [13] this can be expressed by the following first-order update rules:

on insert a into U **update** P **as** $(\neg U(a) \wedge \neg P) \vee (U(a) \wedge P)$
on delete a from U **update** P **as** $(U(a) \wedge \neg P) \vee (\neg U(a) \wedge P)$

[1] This bit is preserved if a change re-inserts an element that already is in U, or tries to delete an element that is not in U.

28th EACSL Annual Conference on Computer Science Logic (CSL 2020).
Editors: Maribel Fernández and Anca Muscholl; Article No. 37; pp. 37:1–37:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

This simple program proves that Parity is in the dynamic complexity class DynFO which contains all queries that can be maintained via first-order formulas that use (and update) some additional stored auxiliary relations.

Motivated by applications in database theory and complexity theory, the class DynFO has been studied extensively in the last three decades. In database theory it is well-known that first-order logic corresponds to the relational core of SQL (see, e.g., [1]). Thus, if a query can be maintained with first-order update rules then, in particular, it can be updated using SQL queries. From a complexity theoretic point of view, first-order logic with built-in arithmetic corresponds to the circuit complexity class uniform AC^0 [3]. Hence queries in DynFO can be evaluated in a highly parallel fashion in dynamic scenarios.

The focus of research on DynFO has been its expressive power. The parity query is a first witness that DynFO is more expressive than FO (the class of queries expressible by first-order formulas in the standard, non-dynamic setting), but it is not the only witness. Further examples include the reachability query for general directed graphs [4], another textbook query that is not in FO but complete for the complexity class NL, which can be characterised (on ordered structures) by the extension of first-order logic with a transitive closure operator. On (classes of) graphs of bounded treewidth, DynFO includes all queries that can be defined in monadic second-order logic [5], which extends first-order logic by quantification over sets. In particular, on strings DynFO also contains all MSO-definable Boolean queries, that is, all regular languages. Actually for strings the update rules do not need any quantifiers [10] proving that regular languages are even in the dynamic complexity class DynProp which is defined via quantifier-free first-order update rules.

These examples show that dynamically first-order logic can, in some cases, sidestep quantifiers and operators which it cannot express statically: parity and set quantifiers, as well as transitive closure operators. Immediately the question arises whether first-order update rules can dynamically maintain *all* queries that are statically expressible in extensions of first-order logic by one of these quantifiers or operators. Note that this does not follow easily, for instance, from the result that the NL-complete reachability query is in DynFO, because the notions of reductions that are available in the dynamic setting are too weak [13].

The extension FO+Parity of first-order logic by parity quantifiers is the natural starting point for a more thorough investigation of how DynFO relates to extensions of FO, as it is arguably the simplest natural extension that extends the expressive power. Unfortunately, however, a result of the form FO+Parity $\subseteq$ DynFO is not in sight[2]. While Parity is in DynFO, already for slightly more complex queries expressible in FO+Parity it seems not to be easy to show that they are in DynFO. In this paper we are particularly interested in the following generalisation of the parity query:

ParityExists: Given a graph whose nodes may be coloured red. Is the number of nodes connected to a red node odd? Edges can be inserted and deleted; nodes can be coloured or uncoloured.

As it is still unknown whether ParityExists is in DynFO, this query is a roadblock for showing that DynFO captures (large subclasses of) FO+Parity. For this reason we study the dynamic complexity of ParityExists. We focus on the following two directions: (1) its relation to the well-understood quantifier-free fragment DynProp of DynFO, and (2) the dynamic complexity of degree-restricted variants.

---

[2] Formally one has to be a little more precise. For technical reasons, one cannot express the query "The size of the domain is even." in DynFO. Therefore we are interested in results of this form for domain-independent queries, that is, queries whose result does not change when isolated elements are added to the domain.

The update rules given above witness that Parity is in DynProp. We show that this is not the case any more for ParityExists.

▶ **Theorem 1.** ParityExists $\notin$ DynProp.

A fine-grained analysis of the quantifier-free complexity is the main contribution of this paper, which also implies Theorem 1. Let $\text{ParityExists}_{\deg \leq k}$ be the variant of the ParityExists query that asks whether the number of nodes that have both an edge to a red node and degree at most k is odd, for some fixed number $k \in \mathbb{N}$.

▶ **Theorem 2.** $\text{ParityExists}_{\deg \leq k}$ *can be maintained in* DynProp *with auxiliary relations of arity* k*, but not with auxiliary relations of arity* $k - 1$*, for any* $k \geq 3$*.*

This result actually has an impact beyond the lower bound given by Theorem 1. It clarifies the structure of DynProp, as it shows that auxiliary relations with higher arities increase the expressive power of quantifier-free update formulas.

Already Dong and Su showed that DynFO has an arity hierarchy [6], i.e., that for each $k \in \mathbb{N}$ there is a query q_k that can be maintained using first-order update rules and k-ary auxiliary relations, but not using $(k-1)$-ary auxiliary relations. The query q_k from [6] is a k-ary query q_k that is evaluated over a $(6k+1)$-ary relation T and returns all k-ary tuples $\bar{a}$ such that the number of $(5k+1)$-ary tuples $\bar{b}$ with $(\bar{a}, \bar{b}) \in T$ is divisible by 4. Dong and Su ask whether the arity of the relation T can be reduced to $3k$, k, or even to 2. Their question for reducing it below $3k$ was motivated by a known reduction of the arity to $3k+1$ [7].

An arity hierarchy for DynProp follows because the query q_k from [6] can be maintained with quantifier-free update rules, though again only for input relations whose arity depends on k. Some progress towards an arity hierarchy for Boolean graph queries was made in [17], where the arities up to $k = 3$ where separated for such queries. If only insertions are allowed, then DynProp is known to have an arity hierarchy for Boolean graph queries [16].

An arity hierarchy for quantifier-free update rules and Boolean graph properties is now an immediate consequence of Theorem 2, in connection with the results for $k \leq 3$ from [17].

▶ **Corollary 3.** DynProp *has a strict arity hierarchy for Boolean graph queries.*

Such an arity hierarchy does *not* exist for DynProp when we consider not graphs as inputs but strings. Gelade et al. show that the class of Boolean queries on strings that are in DynProp are exactly the regular languages, and that every such language can be maintained with binary auxiliary relations [10]. So, relations of higher arity are never necessary in this case.

With respect to DynFO, we cannot answer the question whether ParityExists $\in$ DynFO, but we can generalise the result of Theorem 2 to restrictions beyond fixed numbers k, at least if the update formulas have access to additional built-in relations. Let $\text{ParityExists}_{\deg \leq \log n}$ be the query that asks for the parity of the number of nodes that are connected to a red node and have degree at most $\log n$, where n is the number of nodes of the graph. The binary BIT predicate essentially gives the bit encoding of natural numbers.

▶ **Theorem 4.** $\text{ParityExists}_{\deg \leq \log n}$ *can be maintained in* DynFO *with binary auxiliary relations in the presence of a linear order and* BIT*.*

In particular, the queries $\text{ParityExists}_{\deg \leq k}$, for $k \in \mathbb{N}$, do not induce an arity hierarchy for DynFO. For fixed k, essentially already unary auxiliary relations suffice.

▶ **Theorem 5.** $\text{ParityExists}_{\deg \leq k}$ *can be maintained in* DynFO *with unary auxiliary relations in the presence of a linear order, for every* $k \in \mathbb{N}$*.*

In both results, Theorem 4 and 5, the assumption on the presence of a built-in linear order and the BIT predicate can be lifted when the degree bound of $\text{ParityExists}_{\deg\leq\log n}$ refers to the active domain instead of the whole domain; see Section 4 for a discussion.

Finally, we complement our results by a discussion of how queries expressible in FO extended by arbitrary modulo quantifiers can be maintained in an extension of DynFO. This observation is based on discussions with Samir Datta, Raghav Kulkarni, and Anish Mukherjee.

Outline. After recalling the dynamic descriptive complexity scenario in Section 2, we prove Theorem 2 in Section 3, followed by Theorem 4 and Theorem 5 in Section 4. We conclude in Section 5.

2 Preliminaries: A short introduction to dynamic complexity

We shortly recapitulate the dynamic complexity framework as introduced by Patnaik and Immerman [13], and refer to [15] for details.

In this framework, a (relational, finite) structure $\mathcal{I}$ over some schema σ_{in} can be changed by inserting a tuple into or removing a tuple from a relation of $\mathcal{I}$. A *change* $\alpha = \delta(\bar{a})$ consists of an (abstract) *change operation* δ, which is either INS_R or DEL_R for a relation symbol $R \in \sigma_{\text{in}}$, and a tuple $\bar{a}$ over the domain of $\mathcal{I}$. The change $\text{INS}_R(\bar{a})$ inserts $\bar{a}$ into the relation R of $\mathcal{I}$, and $\text{DEL}_R(\bar{a})$ deletes $\bar{a}$ from that relation. We denote by $\alpha(\mathcal{I})$ the structure that results from applying a change α to the structure $\mathcal{I}$.

A *dynamic program* $\mathcal{P}$ stores an input structure $\mathcal{I}$ as well as an auxiliary structure $\mathcal{A}$ over some auxiliary schema σ_{aux}. For each change operation δ and each auxiliary relation $S \in \sigma_{\text{aux}}$, the dynamic program has a first-order update rule that specifies how S is updated after a change. Each such rule is of the form **on change** $\delta(\bar{p})$ **update** $S(\bar{x})$ **as** $\varphi^S_\delta(\bar{p}, \bar{x})$ where the *update formula* φ^S_δ is over the combined schema $\sigma_{\text{in}} \cup \sigma_{\text{aux}}$ of $\mathcal{I}$ and $\mathcal{A}$. Now, for instance, if a tuple $\bar{a}$ is inserted into an input relation R, the auxiliary relation S is updated to $\{\bar{b} \mid (\mathcal{I}, \mathcal{A}) \models \varphi^S_{\text{INS}_R}(\bar{a}, \bar{b})\}$. In the standard scenario, all relations in both $\mathcal{I}$ and $\mathcal{A}$ are empty initially.

A *k-ary query* q on σ-structures, for some schema σ, maps each σ-structure with some domain D to a subset of D^k, and commutes with isomorphism. A query q is *maintained* by $\mathcal{P}$ if $\mathcal{A}$ has one distinguished relation Ans which, after each sequence of changes, contains the result of q for the current input structure $\mathcal{I}$.

The class DynFO contains all queries that can be maintained by first-order update rules. The class DynProp likewise contains the queries that can be maintained by quantifier-free update rules. We say that a query q is in *k-ary* DynFO (DynProp), for some number $k \in \mathbb{N}$, if it is in DynFO (DynProp) via a dynamic program that uses at most k-ary auxiliary relations.

Sometimes we allow the update formulas to access built-in relations, as for example a predefined linear order $\leq$ and the BIT predicate. We then assume that the input provides a linear order $\leq$, which allows to identify the domain with a prefix of the natural numbers, and a binary relation BIT that contains a tuple (i, j) if the j-th bit in the binary representation of i is 1. Both relations cannot be changed.

For expressibility results we will use the standard scenario from [13] that uses initial input and auxiliary structures with empty relations. Our inexpressibility results are stated for the more powerful scenario where the auxiliary structure is initialised arbitrarily. See also [17] for a discussion of these different scenarios.

Already quantifier-free programs are surprisingly expressive, as they can maintain, for instance, all regular languages [10] and the transitive closure of deterministic graphs [11]. As we have seen in the introduction, also the query PARITY can be maintained by quantifier-free update rules. The following example illustrates a standard technique for maintaining queries with quantifier-free update rules which will also be exploited later.

▶ **Example 6.** For fixed $k \in \mathbb{N}$ let IN-DEG-k be the unary query that, given a graph G, returns the set of nodes with in-degree k. This query is easily definable in FO for each k. We show here that IN-DEG-k can be maintained by a DynProp-program $\mathcal{P}$.

The dynamic program we construct uses *k-lists*, a slight extension of the list technique introduced in [10]. The list technique was used in [17] to maintain emptiness of a unary relation U under insertions and deletions of single elements with quantifier-free formulas. To this end a binary relation LIST which encodes a linked list of the elements in U in the order of their insertion is maintained. Additionally, two unary relations mark the first and the last element of the list. The key insight is that a quantifier-free formula can figure out whether the relation U becomes empty when an element a is deleted by checking whether a is both the first *and* the last element of the list.

To maintain IN-DEG-k the quantifier-free dynamic program $\mathcal{P}$ stores, for every node $v \in V$, a list of all nodes u with $(u, v) \in E$, using a ternary relation LIST_1. More precisely, if $u_1, \ldots, u_m$ are the in-neighbours of v then LIST_1 contains the tuples $(v, u_{i_j}, u_{i_{j+1}})$ where $j_1, \ldots, j_m$ is some permutation of $\{1, \ldots, m\}$. Additionally, the program uses ternary relations $\text{LIST}_2, \ldots, \text{LIST}_k$ such that LIST_i describes paths of length i in the linked list LIST_1. For example, if $(v, u_1, u_2), (v, u_2, u_3)$ and (v, u_3, u_4) are tuples in LIST_1, then $(v, u_1, u_4) \in \text{LIST}_3$. The list LIST_1 comes with $2k$ binary relations $\text{FIRST}_1, \ldots, \text{FIRST}_k, \text{LAST}_1, \cdots, \text{LAST}_k$ that mark, for each $v \in V$, the first and the last k elements of the list of in-neighbours of v, as well as with $k + 2$ unary relations $\text{IS}_0, \cdots, \text{IS}_k, \text{IS}_{>k}$ that count the number of in-neighbours for each $v \in V$ up to k. We call nodes u with $(v, u) \in \text{FIRST}_i$ or $(v, u) \in \text{LAST}_i$ the i-first or the i-last element for v, respectively.

Using these relations, the query can be answered easily: the result is the set of nodes v with $v \in \text{IS}_k$. We show how to maintain the auxiliary relations under insertions and deletions of single edges, and assume for ease of presentation of the update formulas that if a change $\text{INS}_E(u, v)$ occurs then $(u, v) \notin E$ before the change, and a change $\text{DEL}_E(u, v)$ only happens if $(u, v) \in E$ before the change.

Insertions of edges. When an edge (u, v) is inserted, then the node u needs to be inserted into the list of v. This node u also becomes the last element of the list (encoded by a tuple $(v, u) \in \text{LAST}_1$), and the i-last node u' for v becomes the $(i + 1)$-last one, for $i < k$. If only i elements are in the list for v before the change, u becomes the $(i + 1)$-first element for v. The update formulas are as follows:

$$\varphi^{\text{LIST}_i}_{\text{INS}_E}(u, v; x, y, z) \stackrel{\text{def}}{=} \text{LIST}_i(x, y, z) \vee \big(v = x \wedge \text{LAST}_i(x, y) \wedge u = z\big) \qquad \text{for } i \in \{1, \ldots, k\}$$

$$\varphi^{\text{LAST}_1}_{\text{INS}_E}(u, v; x, y) \stackrel{\text{def}}{=} \big(v \neq x \wedge \text{LAST}_1(x, y)\big) \vee \big(v = x \wedge u = y\big)$$

$$\varphi^{\text{LAST}_i}_{\text{INS}_E}(u, v; x, y) \stackrel{\text{def}}{=} \big(v \neq x \wedge \text{LAST}_i(x, y)\big) \vee \big(v = x \wedge \text{LAST}_{i-1}(y)\big) \qquad \text{for } i \in \{2, \ldots, k\}$$

$$\varphi^{\text{FIRST}_i}_{\text{INS}_E}(u, v; x, y) \stackrel{\text{def}}{=} \text{FIRST}_i(x, y) \vee \big(v = x \wedge u = y \wedge \text{IS}_{i-1}(x)\big) \qquad \text{for } i \in \{1, \ldots, k\}$$

$$\varphi^{\text{IS}_0}_{\text{INS}_E}(u, v; x) \stackrel{\text{def}}{=} \big(v \neq x \wedge \text{IS}_0(x)\big)$$

$$\varphi^{\text{IS}_i}_{\text{INS}_E}(u, v; x) \stackrel{\text{def}}{=} \big(v \neq x \wedge \text{IS}_i(x)\big) \vee \big(v = x \wedge \text{IS}_{i-1}(x)\big) \qquad \text{for } i \in \{1, \ldots, k\}$$

$$\varphi^{\text{IS}_{>k}}_{\text{INS}_E}(u, v; x) \stackrel{\text{def}}{=} \text{IS}_{>k}(x) \vee \big(v = x \wedge \text{IS}_k(x)\big)$$

Deletions of edges. When an edge (u, v) is deleted, the hardest task for quantifier-free update formulas is to determine whether, if the in-degree of v was *at least* $k + 1$ before the change, the in-degree of v is now *exactly* k. We use that if an element u is the j-first and at the same time the j'-last element for v, then the list for v contains exactly $j + j' - 1$ elements. If u is removed from the list, $j + j' - 2$ elements remain. So, using the relations $\textsc{First}_j$ and $\textsc{Last}_{j'}$, the exact number m of elements after the change can be determined, if $m \leq 2k - 2$. The relations $\textsc{First}_i$ (and, symmetrically the relations $\textsc{Last}_i$) can be maintained using the relations $\textsc{List}_j$: if the i'-first element u is removed from the list for v, u' becomes the i-first element for $i' \leq i \leq k$ if $(v, u, u') \in \textsc{List}_{i-i'+1}$. The update formulas exploit these insights:

$$\begin{aligned}
\varphi^{\textsc{List}_i}_{\textsc{del}_E}(u, v; x, y, z) &\stackrel{\text{def}}{=} (v \neq x) \wedge \textsc{List}_i(x, y, z) \\
&\vee \big(v = x \wedge u \neq y \wedge \bigwedge_{i' \leq i} \neg\textsc{List}_{i'}(x, y, u) \wedge \textsc{List}_i(x, y, z)\big) \\
&\vee \big(v = x \wedge \bigvee_{\substack{j,j' \\ j+j'=i+1}} \textsc{List}_j(x, y, u) \wedge \textsc{List}_{j'}(x, u, z)\big) \qquad \text{for } i \in \{1, \dots, k\} \\
\varphi^{\textsc{Last}_i}_{\textsc{del}_E}(u, v; x, y) &\stackrel{\text{def}}{=} \big(v \neq x \wedge \textsc{Last}_i(x, y)\big) \\
&\vee \Big(v = x \wedge \bigwedge_{i' \leq i} \neg\textsc{Last}_{i'}(u) \wedge \textsc{Last}_i(y)\Big) \\
&\vee \Big(v = x \wedge \bigvee_{i' \leq i} \big(\textsc{Last}_{i'}(u) \wedge \textsc{List}_{i-i'+1}(x, y, u)\big)\Big) \qquad \text{for } i \in \{1, \dots, k\} \\
\varphi^{\textsc{First}_i}_{\textsc{del}_E}(u, v; x, y) &\stackrel{\text{def}}{=} \big(v \neq x \wedge \textsc{First}_i(x, y)\big) \\
&\vee \Big(v = x \wedge \bigwedge_{i' \leq i} \neg\textsc{First}_{i'}(u) \wedge \textsc{First}_i(y)\Big) \\
&\vee \Big(v = x \wedge \bigvee_{i' \leq i} \big(\textsc{First}_{i'}(u) \wedge \textsc{List}_{i-i'+1}(x, u, y)\big)\Big) \qquad \text{for } i \in \{1, \dots, k\} \\
\varphi^{\textsc{Is}_i}_{\textsc{del}_E}(u, v; x) &\stackrel{\text{def}}{=} \big(v \neq x \wedge \textsc{Is}_i(x)\big) \\
&\vee \big(v = x \wedge \bigvee_{\substack{j,j' \\ j+j'-2=i}} \textsc{First}_j(x, u) \wedge \textsc{Last}_{j'}(x, u)\big) \qquad \text{for } i \in \{0, \dots, k\} \\
\varphi^{\textsc{Is}_{>k}}_{\textsc{del}_E}(u, v; x) &\stackrel{\text{def}}{=} \big(v \neq x \wedge \textsc{Is}_{>k}(x)\big) \\
&\vee \Big(v = x \wedge \textsc{Is}_{>k}(x) \wedge \bigwedge_{\substack{j,j' \\ j+j'-2=k}} \big(\neg\textsc{First}_j(x, u) \vee \neg\textsc{Last}_{j'}(x, u)\big)\Big)
\end{aligned}$$

3 ParityExists and quantifier-free updates

In this section we start our examination of the $\textsc{ParityExists}$ query in the context of quantifier-free update rules. Let us first formalize the query. It is evaluated over *coloured graphs*, that is, directed graphs (V, E) with an additional unary relation R that encodes a set of (red-)*coloured* nodes.[3] A node w of such a graph is said to be *covered* if there is a coloured node $v \in R$ with $(v, w) \in E$. The query $\textsc{ParityExists}$ asks, given a coloured graph, whether the number of covered nodes is odd.

[3] We note that the additional relation R is for convenience of exposition. All our results are also valid for pure graphs: instead of using the relation R one could consider a node v coloured if it has a self-loop $(v, v) \in E$.

As stated in the introduction, ParityExists cannot be maintained with quantifier-free update rules. A closer examination reveals a close connection between a variant of this query and the arity structure of DynProp. Let k be a natural number. The variant $\text{ParityExists}_{\text{deg}\leq k}$ of ParityExists asks whether the number of covered nodes that additionally have in-degree at most k is odd. Note that $\text{ParityExists}_{\text{deg}\leq k}$ is a query on general coloured graphs, not only on graphs with bounded degree.

▶ **Theorem 2.** $\text{ParityExists}_{\text{deg}\leq k}$ *can be maintained in* DynProp *with auxiliary relations of arity k, but not with auxiliary relations of arity $k-1$, for any $k \geq 3$.*

We repeat two immediate consequences which have already been stated in the introduction.

▶ **Theorem 1.** ParityExists $\notin$ DynProp.

▶ **Corollary 3.** DynProp *has a strict arity hierarchy for Boolean graph queries.*

Proof. For every $k \geq 1$ we give a Boolean graph query that can be maintained using k-ary auxiliary relations, but not with $(k-1)$-ary relations.

For $k \geq 3$, we choose the query $\text{ParityExists}_{\text{deg}\leq k}$ which satisfies the conditions by Theorem 2.

For $k = 2$, already [17, Proposition 4.10] shows that the query s-t-TwoPath which asks whether there exists a path of length 2 between two distinguished vertices s and t separates unary DynProp from binary DynProp.

For $k = 1$, we consider the Boolean graph query ParityDegreeDiv3 that asks whether the number of nodes whose degree is divisible by 3 is odd. This query can easily be maintained in DynProp using only unary auxiliary relations. In a nutshell, a dynamic program can maintain for each node v the degree of v modulo 3. So, it maintains three unary relations M_0, M_1, M_2 with the intention that $v \in M_i$ if the degree of v is congruent to i modulo 3. These relations can easily be updated under edge insertions and deletions. Similar as for Parity, a bit P that gives the parity of $|M_0|$ can easily be maintained.

On the other hand, ParityDegreeDiv3 cannot be maintained in DynProp using nullary auxiliary relations. Suppose, towards a contradiction, that it can be maintained by some dynamic program $\mathcal{P}$ that only uses nullary auxiliary relations, and consider an input instance that contains five node $V = \{u_1, u_2, v_1, v_2, v_3\}$ as well as edges $E = \{(u_1, v_1), (u_1, v_2), (u_2, v_1)\}$. No matter the auxiliary database, $\mathcal{P}$ needs to give the same answer after the changes $\alpha_1 \stackrel{\text{def}}{=} \text{INS}_E(u_1, v_3)$ and $\alpha_2 \stackrel{\text{def}}{=} \text{INS}_E(u_2, v_3)$, as it cannot distinguish these tuples using quantifier-free first-order formulas. But α_1 leads to a yes-instance for ParityDegreeDiv3, and α_2 does not. So, $\mathcal{P}$ does not maintain ParityDegreeDiv3. ◀

The rest of this section is devoted to the proof of Theorem 2. First, in Subsection 3.1, we show that $\text{ParityExists}_{\text{deg}\leq k}$ can be maintained with k-ary auxiliary relations, for $k \geq 3$. Here we employ the list technique introduced in Example 6. Afterwards, in Subsection 3.2, we prove that auxiliary relations of arity $k-1$ do not suffice. This proof relies on a known tool for proving lower bounds for DynProp that exploits upper and lower bounds for Ramsey numbers [16].

3.1 Maintaining $\text{ParityExists}_{\text{deg}\leq k}$

We start by proving that $\text{ParityExists}_{\text{deg}\leq k}$ can be maintained in DynProp using k-ary auxiliary relations. In Subsection 3.2 we show that this arity is optimal.

▶ **Proposition 7.** *For every $k \geq 3$,* $\text{ParityExists}_{\text{deg}\leq k}$ *is in k-ary* DynProp.

In the following proof, we write $[n]$ for the set $\{1, \ldots, n\}$ of natural numbers.

Proof. Let $k \geq 3$ be some fixed natural number. We show how a DynProp-program $\mathcal{P}$ can maintain $\text{ParityExists}_{\deg \leq k}$ using at most k-ary auxiliary relations.

The idea is as follows. Whenever a formerly uncoloured node v gets coloured, a certain number $c(v)$ of nodes become covered: v has edges to all these nodes, but no other coloured node has. Because the number $c(v)$ can be arbitrary, the program $\mathcal{P}$ necessarily has to store for each uncoloured node v the parity of $c(v)$ to update the query result. But this is not sufficient. Suppose that another node v' is coloured by a change and that, as a result, a number $c(v')$ of nodes become covered, because they have an edge from v' and so far no incoming edge from another coloured neighbour. Some of these nodes, say, $c(v, v')$ many, also have an incoming edge from v. Of course these nodes do not *become* covered any more when afterwards v is coloured, because they *are* already covered. So, whenever a node v' gets coloured, the program $\mathcal{P}$ needs to update the (parity of the) number $c(v)$, based on the (parity of the) number $c(v, v')$. In turn, the (parity of the) latter number needs to be updated whenever another node v'' is coloured, using the (parity of the) analogously defined number $c(v, v', v'')$, and so on.

It seems that this reasoning does not lead to a construction idea for a dynamic program, as information for more and more nodes needs to be stored, but observe that only those covered nodes are relevant for the query that have in-degree at most k. So, a number $c(v_1, \ldots, v_k)$ does not need to be updated when some other node v_{k+1} gets coloured, because no relevant node has edges from all nodes $v_1, \ldots, v_{k+1}$.

We now present the construction in more detail. A node w is called *active* if its in-degree in-deg(w) is at most k. Let $A = \{a_1, \ldots, a_\ell\}$ be a set of coloured nodes and let $B = \{b_1, \ldots, b_m\}$ be a set of uncoloured nodes, with $\ell + m \leq k$. By $\mathcal{N}_G^{\bullet\circ}(A, B)$ we denote the set of active nodes w of the coloured graph G whose coloured (in-)neighbours are exactly the nodes in A and that have (possibly amongst others) the nodes in B as uncoloured (in-)neighbours. So, $w \in \mathcal{N}_G^{\bullet\circ}(A, B)$ if (1) $\text{in-deg}(w) \leq k$, (2) $(v, w) \in E$ for all $v \in A \cup B$, and (3) there is no edge $(v', w) \in E$ from a coloured node $v' \in R$ with $v' \notin A$. We omit the subscript G and just write $\mathcal{N}^{\bullet\circ}(A, B)$ if the graph G is clear from the context. The dynamic program $\mathcal{P}$ maintains the parity of $|\mathcal{N}_G^{\bullet\circ}(A, B)|$ for all such sets A, B.

Whenever a change $\alpha = \text{INS}_R(v)$ colours a node v of G, the update is as follows. We distinguish the three cases (1) $v \in A$, (2) $v \in B$ and (3) $v \notin A \cup B$. In case (1), the set $\mathcal{N}_{\alpha(G)}^{\bullet\circ}(A, B)$ equals the set $\mathcal{N}_G^{\bullet\circ}(A \setminus \{v\}, B \cup \{v\})$, and the existing auxiliary information can be copied. In case (2), actually $\mathcal{N}_{\alpha(G)}^{\bullet\circ}(A, B) = \emptyset$, as B contains a coloured node. The parity of the cardinality 0 of $\emptyset$ is even. For case (3) we distinguish two further cases. If $|A \cup B| = k$, no active node w can have incoming edges from every node in $A \cup B \cup \{v\}$ as w has in-degree at most k, so $\mathcal{N}_{\alpha(G)}^{\bullet\circ}(A, B) = \mathcal{N}_G^{\bullet\circ}(A, B)$ and the existing auxiliary information is taken over. If $|A \cup B| < k$, then $\mathcal{N}_{\alpha(G)}^{\bullet\circ}(A, B) = \mathcal{N}_G^{\bullet\circ}(A, B) \setminus \mathcal{N}_G^{\bullet\circ}(A, B \cup \{v\})$ and $\mathcal{P}$ can combine the existing auxiliary information.

When a change $\alpha = \text{DEL}_R(v)$ uncolours a node v of G, the necessary updates are symmetrical. The case $v \in A$ is similar to case (2) above: $\mathcal{N}_{\alpha(G)}^{\bullet\circ}(A, B) = \emptyset$, because A contains an uncoloured node. The case $v \in B$ is handled similarly as case (1) above, as we have $\mathcal{N}_{\alpha(G)}^{\bullet\circ}(A, B) = \mathcal{N}_G^{\bullet\circ}(A \cup \{v\}, B \setminus \{v\})$. The third case $v \notin A \cup B$ is treated analogously as case (3) above, but in the sub-case $|A \cup B| < k$ we have that $\mathcal{N}_{\alpha(G)}^{\bullet\circ}(A, B) = \mathcal{N}_G^{\bullet\circ}(A, B) \cup \mathcal{N}_G^{\bullet\circ}(A \cup \{v\}, B)$.

Edge insertions and deletions are conceptionally easy to handle, as they change the sets $\mathcal{N}^{\bullet\circ}(A, B)$ by at most one element. Given all nodes of A and B and the endpoints of the changed edge as parameters, quantifier-free formulas can easily determine whether this is the case for specific sets A, B.

We now present $\mathcal{P}$ formally. For every $\ell \leq k+1$ the program maintains unary relations N_ℓ and $N_\ell^\bullet$ with the indented meaning that for a node w it holds $w \in N_\ell$ if in-deg$(w) = \ell$ and $w \in N_\ell^\bullet$ if w has exactly ℓ coloured in-neighbours. These relations can be maintained as presented in Example 6, requiring some additional, ternary auxiliary relations. We also use a relation $\textsc{Active} \stackrel{\text{def}}{=} N_1 \cup \cdots \cup N_k$ that contains all active nodes with at least one edge.

For every $\ell, m \geq 0$ with $1 \leq \ell + m \leq k$ the programs maintains $(\ell+m)$-ary auxiliary relations $P_{\ell,m}$ with the intended meaning that a tuple $(a_1, \ldots, a_\ell, b_1, \ldots, b_m)$ is contained in $P_{\ell,m}$ if and only if

- the nodes $a_1, \ldots, a_\ell, b_1, \ldots, b_m$ are pairwise distinct,
- $a_i \in R$ and $b_j \notin R$ for $i \in [\ell], j \in [m]$, and
- the set $\mathcal{N}^{\bullet\circ}(A, B)$ has an odd number of elements, where $A = \{a_1, \ldots, a_\ell\}$ and $B = \{b_1, \ldots, b_m\}$.

The following formula $\theta_{\ell,m}$ checks the first two conditions:

$$\theta_{\ell,m}(x_1, \ldots, x_\ell, y_1, \ldots, y_m) \stackrel{\text{def}}{=} \bigwedge_{i \neq j \in [\ell]} x_i \neq x_j \wedge \bigwedge_{i \neq j \in [m]} y_i \neq y_j \wedge \bigwedge_{i \in [\ell]} R(x_i) \wedge \bigwedge_{i \in [m]} \neg R(y_i)$$

Of course, $\mathcal{P}$ also maintains the Boolean query relation $\textsc{Ans}$.

We now describe the update formulas of $\mathcal{P}$ for the relations $P_{\ell,m}$ and $\textsc{Ans}$, assuming that each change actually alters the input graph, so, for example, no changes $\textsc{ins}_E(v, w)$ occur such that the edge (v, w) already exists.

Let $\varphi \oplus \psi \stackrel{\text{def}}{=} (\varphi \wedge \neg\psi) \vee (\neg\varphi \wedge \psi)$ denote the Boolean exclusive-or connector.

Colouring a node v. A change $\textsc{ins}_R(v)$ increases the total number of active, covered nodes by the number of active nodes that have so far no coloured in-neighbour, but an edge from v. That is, this number is increased by $|\mathcal{N}^{\bullet\circ}(\emptyset, \{v\})|$. The update formula for $\textsc{Ans}$ is therefore

$$\varphi^{\textsc{Ans}}_{\textsc{ins}_R}(v) \stackrel{\text{def}}{=} \textsc{Ans} \oplus P_{0,1}(v).$$

We only spell out the more interesting update formulas for the relations $P_{\ell,m}$, for different values of ℓ, m. These formulas list the conditions for tuples $\bar{a} = a_1, \ldots, a_\ell$ and $\bar{b} = b_1, \ldots, b_m$ that $\mathcal{N}^{\bullet\circ}(\{a_1, \ldots, a_\ell\}, \{b_1, \ldots, b_m\})$ is of odd size after a change. The other update formulas are simple variants.

$$\begin{aligned}
&\varphi^{P_{\ell,m}}_{\textsc{ins}_R}(v; x_1, \ldots, x_\ell, y_1, \ldots, y_m) \stackrel{\text{def}}{=} \\
&\quad \bigvee_{i \in [\ell]} \big(v = x_i \wedge P_{\ell-1,m+1}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_\ell, \bar{y}, v)\big) \\
&\quad \vee \Big(\bigwedge_{i \in [\ell]} v \neq x_i \wedge \bigwedge_{i \in [m]} v \neq y_i \wedge \big(P_{\ell,m}(\bar{x}, \bar{y}) \oplus P_{\ell,m+1}(\bar{x}, \bar{y}, v)\big)\Big) \qquad \text{for } \ell \geq 1, \ell + m < k \\
&\varphi^{P_{\ell,m}}_{\textsc{ins}_R}(v; x_1, \ldots, x_\ell, y_1, \ldots, y_m) \stackrel{\text{def}}{=} \\
&\quad \bigvee_{i \in [\ell]} \big(v = x_i \wedge P_{\ell-1,m+1}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_\ell, \bar{y}, v)\big) \\
&\quad \vee \Big(\bigwedge_{i \in [\ell]} v \neq x_i \wedge \bigwedge_{i \in [m]} v \neq y_i \wedge P_{\ell,m}(\bar{x}, \bar{y})\Big) \qquad \text{for } \ell \geq 1, \ell + m = k
\end{aligned}$$

Uncolouring a node v. The update formulas for a change $\textsc{del}_R(v)$ are analogous to the update formulas for a change $\textsc{ins}_R(v)$ as seen above; they are provided in the full version.

Inserting an edge (v, w). When an edge (v, w) is inserted, the number of active, covered nodes can change at most by one. At first, a covered node w might become inactive. This happens when w had in-degree k before the insertion. Or, an active node w becomes covered. This happens if v is coloured and w had no coloured in-neighbour and in-degree at most $k-1$ before the change. The update formula for Ans is accordingly

$$\varphi^{\text{Ans}}_{\text{INS}_E}(v, w) \stackrel{\text{def}}{=} \text{Ans} \oplus \Big(\big(N_k(w) \wedge \bigvee_{i\in[k]} N^\bullet_i(w)\big) \vee \big(R(v) \wedge N^\bullet_0(w) \wedge \bigvee_{i\in[k]} N_{i-1}(w)\big)\Big).$$

The necessary updated for relations $P_{\ell,m}$ are conceptionally very similar. We list the conditions that characterize whether the membership of w in $\mathcal{N}^{\bullet\circ}(A, B)$ changes, for a set $A = \{x_1, \dots, x_\ell\}$ of coloured nodes and a set $B = \{y_1, \dots, y_m\}$ of uncoloured nodes.

- Before the change, $w \in \mathcal{N}^{\bullet\circ}(A, B)$ holds, but not afterwards. This is either because w becomes inactive or because the new edge (v, w) connects w with another coloured node v. This case is expressed by the formula

$$\psi_1 \stackrel{\text{def}}{=} \bigwedge_{i\in[\ell]} E(x_i, w) \wedge N^\bullet_\ell(w) \wedge \bigwedge_{i\in[m]} E(y_i, w) \wedge \big(N_k(w) \vee R(v)\big).$$

- Before the change, $w \in \mathcal{N}^{\bullet\circ}(A, B)$ does not hold, but it does afterwards. Then w needs to be active and to have an incoming edge from all but one node from $A \cup B$, and v is that one node. Additionally, w has no other coloured in-neighbours. The following formulas ψ_2, ψ_3 express these conditions for the cases $v \in A$ and $v \in B$, respectively.

$$\psi_2 \stackrel{\text{def}}{=} \bigvee_{i\in[\ell]} \big(v = x_i \wedge \bigwedge_{j\in[\ell]\setminus\{i\}} E(x_j, w) \wedge \bigwedge_{j\in[m]} E(y_j, w) \wedge N^\bullet_{\ell-1}(w) \wedge \bigvee_{j\in[k]} N_{j-1}(w)\big)$$

$$\psi_3 \stackrel{\text{def}}{=} \bigvee_{i\in[m]} \big(v = y_i \wedge \bigwedge_{j\in[\ell]\setminus\{i\}} E(y_j, w) \wedge \bigwedge_{j\in[\ell]} E(x_j, w) \wedge N^\bullet_{\ell}(w) \wedge \bigvee_{j\in[k]} N_{j-1}(w)\big)$$

The update formula for $P_{\ell,m}$ is then

$$\varphi^{P_{\ell,m}}_{\text{INS}_E}(v, w; x_1, \dots, x_\ell, y_1, \dots, y_m) \stackrel{\text{def}}{=} \theta_{\ell,m}(\bar{x}, \bar{y}) \wedge \big(P_{\ell,m}(\bar{x}, \bar{y}) \oplus (\psi_1 \vee \psi_2 \vee \psi_3)\big).$$

Deleting an edge (v, w). The ideas to construct the update formulas for changes $\text{DEL}_E(v, w)$ are symmetrical to the constructions for changes $\text{INS}_E(v, w)$. When an edge (v, w) is deleted, the node w becomes active if its in-degree before the change was $k + 1$. It is (still) covered, and then is a new active and covered node, if it has coloured in-neighbours other than v. This is the case if w has at least two coloured in-neighbours before the change, or if it has at least one coloured in-neighbour and v is not coloured.

On the other hand, if v was the only coloured in-neighbour of an active node w, this node is not covered any more.

Details are provided in the full version. ◀

Our proof does not go through for $k < 3$, as we use ternary auxiliary relations to maintain whether a node has degree at most k, see Example 6.

3.2 Inexpressibility results for ParityExists$_{\deg\leq k}$

In this subsection we prove that k-ary auxiliary relations are not sufficient to maintain ParityExists$_{\deg\leq k+1}$, for every $k \in \mathbb{N}$. The proof technique we use, and formalise as Lemma 8, is a reformulation of the proof technique of [16], which combines techniques from

[10] and [17] with insights regarding upper and lower bounds for Ramsey numbers. We actually use a special case of the formalisation from [14, Lemma 7.4], which is sufficient for our application.

The technique consists of a sufficient condition under which a Boolean query q cannot be maintained in DynProp with at most k-ary auxiliary relations. The condition basically requires that for each collection $\mathcal{B}$ of subsets of size $k+1$ of a set $\{1, \dots, n\}$, for an arbitrary n, there is a structure $\mathcal{I}$ and a sequence $\alpha(x_1), \dots, \alpha(x_{k+1})$ of changes such that (1) the elements $1, \dots, n$ cannot be distinguished by quantifier-free formulas, and (2) the structure that results from $\mathcal{I}$ by applying the changes $\alpha(i_1), \dots, \alpha(i_{k+1})$ in that order is a positive instance for q exactly if $\{i_1, \dots, i_{k+1}\} \in \mathcal{B}$.

In the following, we denote the set $\{1, \dots, n\}$ by $[n]$ and write $(\mathcal{I}, \bar{a}) \equiv_0 (\mathcal{I}, \bar{b})$ if $\bar{a}$ and $\bar{b}$ have the same length and agree on their quantifier-free type in $\mathcal{I}$, that is, $\mathcal{I} \models \psi(\bar{a})$ if and only if $\mathcal{I} \models \psi(\bar{b})$ for all quantifier-free formulas ψ. We denote the set of all subsets of size k of a set A by $\binom{A}{k}$.

▶ **Lemma 8** ([14]). *Let q be a Boolean query of σ-structures. Then q is not in k-ary DynProp, even with arbitrary initialisation, if for each $n \in \mathbb{N}$ and all subsets $\mathcal{B} \subseteq \binom{[n]}{k+1}$ there exist*

- *a σ-structure $\mathcal{I}$ and a set $P = \{p_1, \dots, p_n\}$ of distinct elements such that*
  - *P is a subset of the domain of $\mathcal{I}$,*
  - *$(\mathcal{I}, p_{i_1}, \dots, p_{i_{k+1}}) \equiv_0 (\mathcal{I}, p_{j_1}, \dots, p_{j_{k+1}})$ for all strictly increasing sequences $i_1, \dots, i_{k+1}$ and $j_1, \dots, j_{k+1}$ over $[n]$, and*
- *a sequence $\alpha(x_1), \dots, \alpha(x_{k+1})$ of changes*

such that for all strictly increasing sequences $i_1, \dots, i_{k+1}$ over $[n]$:

$$(\alpha(p_{i_1}) \circ \dots \circ \alpha(p_{i_{k+1}}))(\mathcal{I}) \in q \iff \{i_1, \dots, i_{k+1}\} \in \mathcal{B}.$$

With the help of Lemma 8 we can show the desired inexpressibility result.

▶ **Proposition 9.** *For every $k \geq 0$, $\textsc{ParityExists}_{\text{deg}\leq k+1}$ is not in k-ary DynProp, even with arbitrary initialisation.*

In the following, for a graph $G = (V, E)$ and some set $X \subseteq V$ of nodes we write $\mathcal{N}^{\rightarrow}(X)$ for the set $\{v \mid \exists u \in X \colon E(u, v)\}$ of out-neighbours of nodes in X. For singleton sets $X = \{x\}$ we just write $\mathcal{N}^{\rightarrow}(x)$ instead of $\mathcal{N}^{\rightarrow}(\{x\})$.

Proof. Let $k \in \mathbb{N}$ be fixed. We apply Lemma 8 to show that $\textsc{ParityExists}_{\text{deg}\leq k+1}$ is not in k-ary DynProp.

The basic proof idea is simple. Given a collection $\mathcal{B} \subseteq \binom{[n]}{k+1}$, we construct a graph $G = (V, E)$ with distinguished nodes $P = \{p_1, \dots, p_n\} \subseteq V$ such that (1) each node has in-degree at most $k+1$ and (2) for each $B \in \binom{[n]}{k+1}$ the set $\mathcal{N}^{\rightarrow}(\{p_i \mid i \in B\})$ is of odd size if and only if $B \in \mathcal{B}$. Then applying a change sequence α which colours all nodes p_i with $i \in B$ to G results in a positive instance of $\textsc{ParityExists}_{\text{deg}\leq k+1}$ if and only if $B \in \mathcal{B}$. An invocation of Lemma 8 yields the intended lower bound.

It remains to construct the graph G. Let S be the set of all non-empty subsets of $[n]$ of size at most $k+1$. We choose the node set V of G as the union of P and S. Only nodes in P will be coloured, and only nodes from S will be covered. A first attempt to realise the idea mentioned above might be to consider an edge set $E_{k+1} \stackrel{\text{def}}{=} \{(p_i, B) \mid B \in \mathcal{B}, i \in B\}$: then, having fixed some set $B \in \mathcal{B}$, the node B becomes covered whenever the nodes p_i with $i \in B$ are coloured. However, also some nodes $B' \neq B$ will be covered, namely if $B' \cap B \neq \emptyset$, and the number of these nodes influences the query result. We ensure that

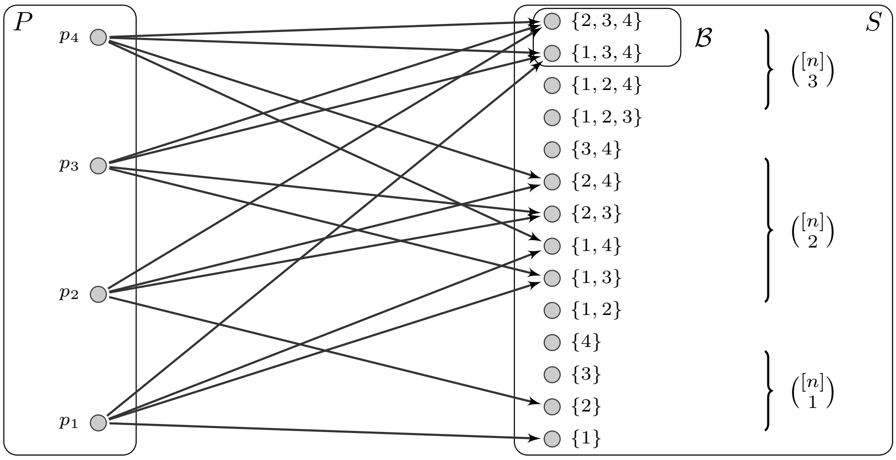

Figure 1 Example for the construction in the proof of Proposition 9, with $k = 2$ and $n = 4$.

the set of nodes $B' \neq B$ that are covered by $\{p_i \mid i \in B\}$ is of even size, so that the parity of $|\mathcal{N}^{\rightarrow}(\{p_i \mid i \in B\})|$ is determined by whether $B \in \mathcal{B}$ holds. This will be achieved by introducing edges to nodes $\binom{[n]}{i} \in S$ for $i \leq k$ such that for every subset P' of P of size at most k the number of nodes from S that have an incoming edge from *all* nodes from P' is even. By an inclusion-exclusion argument we conclude that for any set $\hat{P} \in \binom{P}{k+1}$ the number of nodes from S that have an incoming edge from *some* node of $\hat{P}$, but not from all of them, is even. It follows that whenever $k+1$ nodes $p_{i_1}, \ldots, p_{i_{k+1}}$ are marked, the number of covered nodes is odd precisely if there is one node in S that has an edge from *all* nodes $p_{i_1}, \ldots, p_{i_{k+1}}$, which is the case exactly if $\{i_1, \ldots, i_{k+1}\} \in \mathcal{B}$.

We now make this precise. Let n be arbitrary and let $P = \{p_1, \ldots, p_n\}$. For a set $X \subseteq [n]$ we write P_X for the set $\{p_i \mid i \in X\}$.

The structure $\mathcal{I}$ we construct consists of a coloured graph $G = (V, E)$ with nodes $V \stackrel{\text{def}}{=} P \cup S$, where $S \stackrel{\text{def}}{=} \binom{[n]}{1} \cup \cdots \cup \binom{[n]}{k+1}$, and initially empty set $R \stackrel{\text{def}}{=} \emptyset$ of coloured nodes. The edge set $E = E_1 \cup \cdots \cup E_{k+1}$ is constructed iteratively in $k+1$ steps. We first define the set E_{k+1} and define the set E_j based on the set $E_{>j} \stackrel{\text{def}}{=} \bigcup_{j'=j+1}^{k+1} E_{j'}$.

The set E_{k+1} consists of all edges (p_i, B) such that $B \in \mathcal{B}$ and $i \in B$. For the construction of the set E_j with $j \in \{1, \ldots, k\}$ we assume that all sets $E_{j'}$ with $j' > j$ have already been constructed. Let $X \in \binom{[n]}{j}$ be a set and let m be the number of nodes $Y \in S$ for which there are already edges $(p_i, Y) \in E_{>j}$ for all nodes p_i in P_X. If m is odd, then there is so far an odd number of nodes from S that have an incoming edge from all $p_i \in P_X$. As we want this number to be even, we let E_j contain edges (p_i, X) for all $i \in X$. If m is even, no edges are added to E_j. See Figure 1 for an example of this construction. Note that for each $X \in \binom{[n]}{i}$, for $i \in \{1, \ldots, k+1\}$, the degree of X in G is at most i, and therefore also at most $k+1$.

We now show that for a set $B \in \binom{[n]}{k+1}$ the cardinality of $\mathcal{N}^{\rightarrow}(P_B)$ is indeed odd if and only if $B \in \mathcal{B}$. This follows by an inclusion-exclusion argument. For a set $X \subseteq [n]$ the set $\mathcal{N}^{\rightarrow}(P_X)$ contains all nodes with an incoming edge from a node in P_X. It is therefore equal to the union $\bigcup_{i \in X} \mathcal{N}^{\rightarrow}(p_i)$. When we sum up the cardinalities of these sets $\mathcal{N}^{\rightarrow}(p_i)$, any node in $\mathcal{N}^{\rightarrow}(P_X)$ with edges to both p_i and p_j, for numbers $i, j \in X$, is accounted for twice. Continuing this argument, the cardinality of $\mathcal{N}^{\rightarrow}(X)$ can be computed as follows.

$$\left|\mathcal{N}^{\rightarrow}(P_X)\right| = \sum_{i \in X} \left|\mathcal{N}^{\rightarrow}(p_i)\right| - \sum_{\substack{i,j \in X \\ i<j}} \left|\mathcal{N}^{\rightarrow}(p_i) \cap \mathcal{N}^{\rightarrow}(p_j)\right| + \cdots + (-1)^{|X|-1} \Big| \bigcap_{i \in X} \mathcal{N}^{\rightarrow}(p_i) \Big|$$

By construction of G, the set $\bigcap_{i \in Y} \mathcal{N}^{\rightarrow}(p_i)$ is of even size, for all sets $Y \subseteq [n]$ of size at most k. Consequently, for each $X \in \binom{[n]}{k+1}$ the parity of $\left|\mathcal{N}^{\rightarrow}(P_X)\right|$ is determined by the parity of $\left|\bigcap_{i \in X} \mathcal{N}^{\rightarrow}(p_i)\right|$, the last term in the above equation. Only the node X can possibly have incoming edges from all nodes p_i in P_X, and these edges exist if and only if $X \in \mathcal{B}$.

Let $\alpha(x_1), \ldots, \alpha(x_{k+1})$ be the change sequence $\text{INS}_R(x_1), \ldots, \text{INS}_R(x_{k+1})$ that colours the nodes $x_1, \ldots, x_{k+1}$. Let $B \in \binom{[n]}{k+1}$ be of the form $\{i_1, \ldots, i_{k+1}\}$ with $i_1 < \cdots < i_{k+1}$. The change sequence $\alpha_B \stackrel{\text{def}}{=} \alpha(p_{i_1}) \cdots \alpha(p_{i_{k+1}})$ results in a graph where the set of coloured nodes is exactly P_B. As all nodes in $\mathcal{N}^{\rightarrow}(P_B)$ have degree at most $k+1$ and the set $\mathcal{N}^{\rightarrow}(P_B)$ is of odd size exactly if $B \in \mathcal{B}$, we have that $\alpha_B(\mathcal{I})$ is a positive instance of $\textsc{ParityExists}_{\deg \leq k+1}$ if and only if $B \in \mathcal{B}$. ◀

4 ParityExists and first-order updates

As discussed in the introduction, the $\textsc{Parity}$ query can be easily maintained with first-order update rules. So far we have seen that its generalisation $\textsc{ParityExists}$ can only be maintained with quantifier-free update rules if the in-degree of covered nodes is bounded by a constant. Now we show that with first-order update rules, this query can be maintained if the in-degree is bounded by $\log n$, where n is the number of nodes in the graph. We emphasise that only the in-degree of covered nodes is bounded, while a coloured node v can cover arbitrarily many nodes. If also the out-degree of coloured node is restricted, maintenance in DynFO becomes trivial.

We start by providing a dynamic program with first-order update rules that maintains $\textsc{ParityExists}_{\deg \leq k}$, for a constant k, and only uses unary relations apart from a linear order. Thus, in contrast to quantifier-free update rules, this query cannot be used to obtain an arity hierarchy for graph queries for first-order update rules. Afterwards we will exploit the technique used here to maintain $\textsc{ParityExists}_{\deg \leq \log n}$ with binary auxiliary relations.

▶ **Theorem 5.** $\textsc{ParityExists}_{\deg \leq k}$ *can be maintained in* DynFO *with unary auxiliary relations in the presence of a linear order, for every* $k \in \mathbb{N}$.

An intuitive reason why quantifier-free dynamic programs need auxiliary relations of growing arity to maintain $\textsc{ParityExists}_{\deg \leq k}$ is that for checking whether some change, for instance the colouring of a node v, is "relevant" for some node w, it needs to have access to all of w's "important" neighbours. Without quantification, the only way to do this is to explicitly list them as elements of the tuple for which the update formula decides whether to include it in the auxiliary relation.

With quantification and a linear order, sets of neighbours can be defined more easily, if the total number of neighbours is bounded by a constant. Let us fix a node w with at most k (in-)neighbours, for some constant k. Thanks to the linear order, the neighbours can be distinguished as first, second, ..., k-th neighbour of w, and any subset of these nodes is uniquely determined and can be defined in FO by the node w and a set $I \subseteq \{1, \ldots, k\}$ that *indexes* the neighbours. With this idea, the proof of Proposition 7 can be adjusted appropriately for Theorem 5.

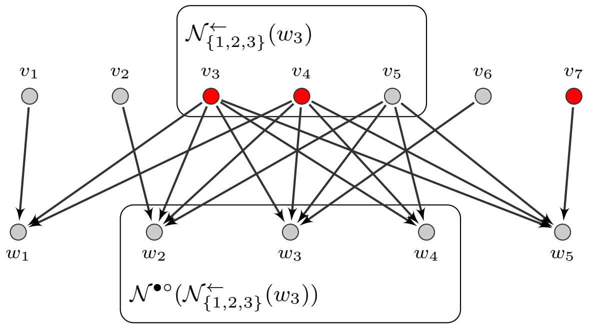

Figure 2 An illustration of the notation used in the proof of Theorem 5. The set $\mathcal{N}^{\bullet\circ}_G(\mathcal{N}^{\leftarrow}_{\{1,2,3\}}(w_3))$ does not include w_1, as there is no edge (v_5, w_1), and it does not include w_5, as there is an edge (v_7, w_5) for a coloured node $v_7 \notin \mathcal{N}^{\leftarrow}_{\{1,2,3\}}(w_3)$.

Proof sketch (of Theorem 5). Let $k \in \mathbb{N}$ be some constant. Again, we call a node *active* if its in-degree is at most k. We sketch a dynamic program that uses a linear order on the nodes and otherwise at most unary auxiliary relations.

Let I be a non-empty subset of $\{1,\dots,k\}$, and let w be an active node with at least $\max(I)$ in-neighbours. The set $\mathcal{N}^{\leftarrow}_I(w)$ of *I-indexed in-neighbours* of w includes a node v if and only if (v, w) is an edge in the input graph and v is the i-th in-neighbour of w with respect to the linear order, for some $i \in I$. The following notation is similar as in the proof of Proposition 7. For a graph G and an arbitrary set C of (coloured and uncoloured) nodes, we denote the set of active nodes that have an incoming edge from every node in C and no coloured in-neighbour that is not in C by $\mathcal{N}^{\bullet\circ}_G(C)$. An example for these notions is depicted in Figure 2.

For every $I \subseteq \{1,\dots,k\}$ with $I \neq \emptyset$ we introduce an auxiliary relation P_I with the following intended meaning. An active node w with at least $\max(I)$ neighbours is in P_I if and only if (1) w has no coloured in-neighbours that are not contained in $\mathcal{N}^{\leftarrow}_I(w)$, and (2) the set $\mathcal{N}^{\bullet\circ}_G(\mathcal{N}^{\leftarrow}_I(w))$ has odd size. Note that (1) implies that $w \in \mathcal{N}^{\bullet\circ}_G(\mathcal{N}^{\leftarrow}_I(w))$.

An auxiliary relation P_I basically replaces the relations $P_{\ell,m}$ with $\ell + m = |I|$ from the proof of Proposition 7, and the updates are mostly analogous.

We explain how the query relation Ans and the relations P_I are updated when a modification to the input graph occurs. When a node v is coloured, the query relation is only changed if v becomes the only coloured neighbour of an odd number of active nodes. This is the case if and only if there actually is an active and previously uncovered node w that v has an edge to and if $w \in P_I$ for the set $I \stackrel{\text{def}}{=} \{i\}$, where i is the number such that v is the i-th in-neighbour of w with respect to the linear order.

The update of a relation P_I after the colouring of a node v is as follows. Let G be the graph before the change is applied, and G' the changed graph. Let w be any active node. If v is an I-indexed in-neighbour of w, no change regarding $w \in P_I$ is necessary. Otherwise, some nodes in $\mathcal{N}^{\bullet\circ}_G(\mathcal{N}^{\leftarrow}_I(w))$ might now have a coloured neighbour v that is not contained in $\mathcal{N}^{\leftarrow}_I(w)$, and therefore are not contained in $\mathcal{N}^{\bullet\circ}_{G'}(\mathcal{N}^{\leftarrow}_I(w))$. Let w' be such a node, that is, a node with an edge from v and every node in $\mathcal{N}^{\leftarrow}_I(w)$, and let I' be such that $\mathcal{N}^{\leftarrow}_{I'}(w') = \mathcal{N}^{\leftarrow}_I(w) \cup \{v\}$. The parity of the number of nodes in $\mathcal{N}^{\bullet\circ}_G(\mathcal{N}^{\leftarrow}_I(w)) \setminus \mathcal{N}^{\bullet\circ}_{G'}(\mathcal{N}^{\leftarrow}_I(w))$ is odd if and only if $w' \in P_{I'}$. This can be used to update P_I.

We do not present the updates for the remaining changes as they can be easily constructed along the same lines. ◀

It is easy to maintain a linear order on the non-isolated nodes of an input graph [8], which is all that is needed for the proof of Theorem 5. So, $\text{ParityExists}_{\deg\leq k}$ can also be maintained in DynFO without a predefined linear order, at the expense of binary auxiliary relations.

Unfortunately we cannot generalise the technique from Theorem 2 for $\text{ParityExists}_{\deg\leq k}$ to ParityExists, but only to $\text{ParityExists}_{\deg\leq \log n}$, which asks for the parity of the number of covered nodes with in-degree at most $\log n$. Here, n is the number of nodes of the graph.

▶ **Theorem 4.** $\text{ParityExists}_{\deg\leq \log n}$ *can be maintained in* DynFO *with binary auxiliary relations in the presence of a linear order and* BIT*.*

Proof sketch. With the help of the linear order we identify the node set V of size n of the input graph with the numbers $\{0, \ldots, n-1\}$, and use BIT to access the bit encoding of these numbers. Any node $v \in V$ then naturally encodes a set $I(v) \subseteq \{1, \ldots, \log n\}$: $i \in \{1, \ldots, \log n\}$ is contained in $I(v)$ if and only if the i-th bit in the bit encoding of v is 1.

The proof of Theorem 5 constructs a dynamic program that maintains unary relations P_I with $I \subseteq \{1, \ldots, k\}$, and $w \in P_I$ holds if $w \in \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))$ and if $|\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))|$ is odd. We replace these relations by a single binary relation P, with the intended meaning that $(v, w) \in P$ if $w \in \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{I(v)}^{\leftarrow}(w))$ and if $|\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{I(v)}^{\leftarrow}(w))|$ is odd.

A dynamic program that maintains $\text{ParityExists}_{\deg\leq \log n}$ can then be constructed along the same lines as in the proof of Theorem 5. ◀

In addition to a linear order, [8] also shows how corresponding relations addition and multiplication can be maintained for the active domain of a structure. As BIT is first-order definable in the presence of addition and multiplication, and vice versa (see e.g. [12, Theorem 1.17]), both a linear order and BIT on the active domain can be maintained, still using only binary auxiliary relations. So, the variant of $\text{ParityExists}_{\deg\leq \log n}$ that considers n to be the number of non-isolated nodes, instead of the number of all nodes, can be maintained in binary DynFO without assuming built-in relations.

5 Conclusion

We studied the dynamic complexity of the query ParityExists as well as its bounded degree variants. While it remains open whether ParityExists is in DynFO, we showed that $\text{ParityExists}_{\deg\leq \log n}$ is in DynFO and that $\text{ParityExists}_{\deg\leq k}$ is in DynProp, for fixed $k \in \mathbb{N}$. The latter result is the basis for an arity hierarchy for DynProp for Boolean graph queries. Several open questions remain.

▶ **Open question.** *Can* ParityExists *be maintained with first-order updates rules? If so, are all (domain-independent) queries from* FO+Parity *also in* DynFO*?*

▶ **Open question.** *Is there an arity hierarchy for* DynFO *for Boolean graph queries?*

Orthogonally to the perspectives taken in this work, one can ask how many auxiliary bits are necessary to maintain the query ParityExists or, more generally, all queries expressible in first-order logic extended by modulo quantifiers. It is convenient to switch the view point from first-order updates to updates computed by AC^0 circuits for discussing the amount of auxiliary bits. The class DynFO corresponds to (uniform) DynAC^0, and allows for polynomially many auxiliary bits. It is not hard to see that if one allows quasi-polynomially many auxiliary bits and update circuits of quasi-polynomial size, then all queries expressible in first-order logic extended by modulo quantifiers can be maintained. This was observed in discussions with Samir Datta, Raghav Kulkarni and Anish Mukherjee. A proof sketch is provided in the full version of this paper.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.

2 Miklós Ajtai. Σ_1^1-Formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983. doi:10.1016/0168-0072(83)90038-6.

3 David A. Mix Barrington, Neil Immerman, and Howard Straubing. On Uniformity within NC^1. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990. doi:10.1016/0022-0000(90)90022-D.

4 Samir Datta, Raghav Kulkarni, Anish Mukherjee, Thomas Schwentick, and Thomas Zeume. Reachability Is in DynFO. *J. ACM*, 65(5):33:1–33:24, August 2018. doi:10.1145/3212685.

5 Samir Datta, Anish Mukherjee, Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. A Strategy for Dynamic Programs: Start over and Muddle through. *Logical Methods in Computer Science*, Volume 15, Issue 2, May 2019. doi:10.23638/LMCS-15(2:12)2019.

6 Guozhu Dong and Jianwen Su. Arity Bounds in First-Order Incremental Evaluation and Definition of Polynomial Time Database Queries. *J. Comput. Syst. Sci.*, 57(3):289–308, 1998. doi:10.1006/jcss.1998.1565.

7 Guozhu Dong and Louxin Zhang. Separating Auxiliary Arity Hierarchy of First-Order Incremental Evaluation Systems Using (3k+1)-ary Input Relations. *Int. J. Found. Comput. Sci.*, 11(4):573–578, 2000. doi:10.1142/S0129054100000302.

8 Kousha Etessami. Dynamic Tree Isomorphism via First-Order Updates. In Alberto O. Mendelzon and Jan Paredaens, editors, *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 235–243. ACM Press, 1998. doi:10.1145/275487.275514.

9 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, Circuits, and the Polynomial-Time Hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. doi:10.1007/BF01744431.

10 Wouter Gelade, Marcel Marquardt, and Thomas Schwentick. The dynamic complexity of formal languages. *ACM Trans. Comput. Log.*, 13(3):19, 2012. doi:10.1145/2287718.2287719.

11 William Hesse. *Dynamic Computational Complexity*. PhD thesis, University of Massachusetts Amherst, 2003.

12 Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999. doi:10.1007/978-1-4612-0539-5.

13 Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel, dynamic complexity class. *J. Comput. Syst. Sci.*, 55(2):199–209, 1997. doi:10.1006/jcss.1997.1520.

14 Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. Dynamic Complexity under Definable Changes. *ACM Trans. Database Syst.*, 43(3):12:1–12:38, 2018. doi:10.1145/3241040.

15 Thomas Schwentick and Thomas Zeume. Dynamic complexity: recent updates. *SIGLOG News*, 3(2):30–52, 2016. doi:10.1145/2948896.2948899.

16 Thomas Zeume. The dynamic descriptive complexity of k-clique. *Inf. Comput.*, 256:9–22, 2017. doi:10.1016/j.ic.2017.04.005.

17 Thomas Zeume and Thomas Schwentick. On the quantifier-free dynamic complexity of Reachability. *Inf. Comput.*, 240:108–129, 2015. doi:10.1016/j.ic.2014.09.011.