

DYNAMIC COMPLEXITY OF PARITY EXISTS QUERIES

NILS VORTMEIER^a AND THOMAS ZEUME^b

^a University of Zurich, Switzerland
e-mail address: nils.vortmeier@uzh.ch

^b Ruhr University Bochum, Germany
e-mail address: thomas.zeume@rub.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.

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 [Ajt83, FSS84]. 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 flipped¹. In the dynamic complexity framework by Patnaik and Immerman [PI97] 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** $(U(a) \wedge \neg P) \vee (\neg U(a) \wedge P)$

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.

Key words and phrases: Dynamic complexity, parity quantifier, arity hierarchy.

¹This 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 .

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., [AHV95]). 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 [BIS90]. 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 one. Further examples include the reachability query for general directed graphs [DKM⁺18], 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 with bounded treewidth, DynFO includes all queries that can be defined in monadic second-order logic MSO [DMS⁺19], which extends first-order logic by quantification over sets. In particular, DynFO contains all MSO -definable Boolean queries on strings, that is, all regular languages. For strings, the first-order update rules do not even need any quantifiers [GMS12], 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 in the dynamic setting, 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 [PI97].

The extension $\text{FO}+\text{Parity}$ 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 $\text{FO}+\text{Parity} \subseteq \text{DynFO}$ is not in sight². While PARITY is in DynFO , already for slightly more complex queries expressible in $\text{FO}+\text{Parity}$ 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.

²Formally one has to be a little more precise. If one does not allow an appropriate initialisation mechanism, one cannot express the query “The size of the domain is even.” in DynFO , which implies $\text{FO}+\text{Parity} \not\subseteq \text{DynFO}$ for this variant of DynFO . However, if the initialisation is powerful enough to answer the query for the initial input, maintenance under changes is trivial, as they are not allowed to change the domain.

We aim to study the ability of dynamic programs to maintain queries under changes, we are less interested in inexpressibility results that crucially rely on weaknesses of the initialisation. Therefore we are interested in results of the form $\text{FO}+\text{Parity} \subseteq \text{DynFO}$ for domain-independent queries, that is, queries whose result does not change when isolated elements are added to the domain. As we only consider initial input structures with empty relations, and so initial inputs only differ in their domain, the initial query answer for domain-independent queries is the same for all possible initial input structures, and therefore trivial to obtain regardless of the power of the initialisation.

As it is still unknown whether `PARITY EXISTS` 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 `PARITY EXISTS`. 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.

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

Theorem 1.1. `PARITY EXISTS` \notin `DynProp`.

A fine-grained analysis of the quantifier-free complexity is the main contribution of this paper, which also implies Theorem 1.1. Let `PARITY EXISTSdeg \leq k` be the variant of the `PARITY EXISTS` 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 1.2. `PARITY EXISTSdeg \leq k` can be maintained in `DynProp` with auxiliary relations of arity k , for any $k \geq 3$, but not with auxiliary relations of arity $k - 1$, even on graphs with in-degree at most k .

This result actually has an impact beyond the lower bound given by Theorem 1.1. It clarifies the structure of `DynProp`, as it shows that auxiliary relations with higher arities increase the expressive power of quantifier-free update formulas even on graph structures.

Already Dong and Su showed that `DynFO` has an arity hierarchy [DS98], 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 given by Dong and Su [DS98] is a k -ary query 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 $3k + 1$ [DZ00].

An arity hierarchy for `DynProp`, though again only for input relations whose arity depends on k , follows from the observation that the query q_k can be maintained with quantifier-free update rules. Some progress towards an arity hierarchy for queries over a fixed schema was made by Zeume and Schwentick [ZS15], who separated the arities up to $k = 3$ for Boolean queries on graphs. If only insertions are allowed, then `DynProp` is known to have an arity hierarchy for Boolean graph queries [Zeu17].

An arity hierarchy for quantifier-free update rules and Boolean graph properties is now an immediate consequence of Theorem 1.2, in connection with the separation results for $k \leq 3$ [ZS15].

Corollary 1.3. `DynProp` has a strict arity hierarchy for Boolean graph queries.

We note that 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 can be maintained in `DynProp` are exactly the regular languages, and that every such language can be maintained with binary auxiliary relations [GMS12]. So, relations of higher arity are never necessary in this case.

With respect to `DynFO`, we cannot answer the question whether `PARITY EXISTS` \in `DynFO`, but we can generalise the upper bound of Theorem 1.2 to restrictions beyond fixed numbers k , at least if the update formulas have access to additional built-in relations. Let

$\text{PARITYEXISTS}_{\text{deg} \leq \log n}$ 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 1.4. $\text{PARITYEXISTS}_{\text{deg} \leq \log n}$ can be maintained in DynFO with binary auxiliary relations in the presence of a linear order and BIT.

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

Theorem 1.5. $\text{PARITYEXISTS}_{\text{deg} \leq k}$ 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 1.4 and 1.5, the assumption on the presence of a built-in linear order and the BIT predicate can be lifted, when, for Theorem 1.4, the degree bound of $\text{PARITYEXISTS}_{\text{deg} \leq \log n}$ refers to the active domain instead of the whole domain, and, for Theorem 1.5, when binary auxiliary relations are allowed. See Section 4 for a more detailed 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 1.2 in Section 3, followed by Theorem 1.4 and Theorem 1.5 in Section 4. Section 5 contains the discussion regarding maintaining PARITYEXISTS and similar queries in extensions of DynFO. We conclude in Section 6.

2. PRELIMINARIES AND A SHORT INTRODUCTION TO DYNAMIC COMPLEXITY

In this article, we write $[n]$ for the set $\{1, \dots, n\}$ of natural numbers.

We now shortly recapitulate the dynamic complexity framework as introduced by Patnaik and Immerman [PI97], and refer to Reference [SZ16] 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} over σ_{in} 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** $\varphi_\delta^S(\bar{p}; \bar{x})$ where the *update formula* φ_δ^S 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 $\{\bar{b} \mid (\mathcal{I}, \mathcal{A}) \models \varphi_{\text{INS}_R}^S(\bar{a}; \bar{b})\}$. In the standard scenario, all relations in both \mathcal{I} and \mathcal{A} are empty initially.

An *m-ary query* q on σ -structures, for some schema σ , maps each σ -structure with some domain D to a subset of D^m , 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`, for some number $k \in \mathbb{N}$, if it is in `DynFO` via a dynamic program that uses at most k -ary auxiliary relations; and likewise for `DynProp`.

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 binary relation that stores a linear order \leq on the domain, 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. Built-in relations are not changed by update rules.

For expressibility results we will use the standard scenario of Patnaik and Immerman [PI97] that uses initial input and auxiliary structures with empty relations. Our inexpressibility results are stated for the more powerful scenario where the auxiliary structure may be initialised arbitrarily. See also Reference [ZS15] for a discussion of these different scenarios.

Already quantifier-free programs are surprisingly expressive, as they can maintain, for instance, all regular languages [GMS12] and the transitive closure of deterministic graphs [Hes03]. 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 2.1. For fixed $k \in \mathbb{N}$, let `SIZE- k` be the Boolean query that asks whether the size of a unary relation `U` is equal to k , so, whether $|\mathbf{U}| = k$ holds. This query is easily definable in `FO` for each k . We show here that `SIZE- k` can be maintained by a `DynProp`-program \mathcal{P} with binary auxiliary relations.

The dynamic program we construct uses k -lists, a slight extension of the list technique introduced by Gelade, Marquardt and Schwentick [GMS12]. The list technique was also used to maintain emptiness of a unary relation `U` under insertions and deletions of single elements with quantifier-free formulas [ZS15]. To this end, a binary relation `LIST` is maintained which encodes a linked list of the elements in `U` in the order of their insertion. 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.

Let us implement this highlevel idea for a fixed $k \in \mathbb{N}$. Let $\ell \stackrel{\text{def}}{=} k + 1$. To maintain `SIZE- k` , the quantifier-free dynamic program \mathcal{P} stores a list of all elements $u \in \mathbf{U}$, using a binary relation `LIST1`. More precisely, if u_1, \dots, u_m are the elements in `U`, then `LIST1` contains the tuples $(u_{i_j}, u_{i_{j+1}})$, for $1 \leq j \leq m - 1$, where i_1, \dots, i_m is some permutation of $\{1, \dots, m\}$. Additionally, the program uses binary relations `LIST2, \dots, LISTℓ` such that `LIST i` describes paths of length i in the linked list `LIST1`. For example, if $(u_1, u_2), (u_2, u_3)$ and (u_3, u_4) are tuples in `LIST1`, then $(u_1, u_4) \in \text{LIST}_3$. The list `LIST1` comes with 2ℓ unary relations `FIRST1, \dots, FIRSTℓ, LAST1, \dots, LASTℓ` that mark the first and the last ℓ elements of the list, as well as with $k + 2$ nullary relations `IS0, \dots, ISk, IS>k` that indicate the number of elements in `U` up to k . We call nodes u with $u \in \text{FIRST}_i$ or $u \in \text{LAST}_i$ the i -first or the i -last element, respectively.

Using these relations, the query can be answered easily: the result is given by `ISk`. We show how to maintain the auxiliary relations under insertions and deletions of single

elements, and assume for ease of presentation of the update formulas that if a change $\text{INS}_U(u)$ occurs then $u \notin U$ before the change, and a change $\text{DEL}_U(u)$ only happens if $u \in U$ before the change.

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

$$\begin{aligned}
\varphi_{\text{INS}_U}^{\text{LIST}_i}(u; x, y) &\stackrel{\text{def}}{=} \text{LIST}_i(x, y) \vee (\text{LAST}_i(x) \wedge u = y) && \text{for } i \in \{1, \dots, \ell\} \\
\varphi_{\text{INS}_U}^{\text{LAST}_1}(u; x) &\stackrel{\text{def}}{=} u = x \\
\varphi_{\text{INS}_U}^{\text{LAST}_i}(u; x) &\stackrel{\text{def}}{=} \text{LAST}_{i-1}(x) && \text{for } i \in \{2, \dots, \ell\} \\
\varphi_{\text{INS}_U}^{\text{FIRST}_i}(u; x) &\stackrel{\text{def}}{=} \text{FIRST}_i(x) \vee (u = x \wedge \text{IS}_{i-1}) && \text{for } i \in \{1, \dots, \ell\} \\
\varphi_{\text{INS}_U}^{\text{IS}_0}(u) &\stackrel{\text{def}}{=} \perp \\
\varphi_{\text{INS}_U}^{\text{IS}_i}(u) &\stackrel{\text{def}}{=} \text{IS}_{i-1} && \text{for } i \in \{1, \dots, k\} \\
\varphi_{\text{INS}_U}^{\text{IS}_{>k}}(u) &\stackrel{\text{def}}{=} \text{IS}_k \vee \text{IS}_{>k}
\end{aligned}$$

Deletions of elements. When an element u is deleted, the hardest task for quantifier-free update formulas is to determine whether, if the size of U was *at least* $k + 1$ before the change, its size is now *exactly* k . We use that if an element u is the j -first and at the same time the j' -last element, then the list contains exactly $j + j' - 1$ elements. If u is removed from the list, $j + j' - 2$ elements remain. So, using the relations FIRST_j and $\text{LAST}_{j'}$, the exact number m of elements after the change can be determined, if $m \leq 2\ell - 2 = 2k$, and in particular, it can be determined whether this number is k . The relations FIRST_i (and, symmetrically the relations LAST_i) can be maintained using the relations LIST_j : if the i' -first element u is removed from the list, then u' becomes the i -first element for $i' \leq i \leq \ell$ if $(u, u') \in \text{LIST}_{i-i'+1}$. The update formulas exploit these insights:

$$\begin{aligned}
\varphi_{\text{DEL}_U}^{\text{LIST}_i}(u; x, y) &\stackrel{\text{def}}{=} (u \neq x \wedge \bigwedge_{i' \leq i} \neg \text{LIST}_{i'}(x, u) \wedge \text{LIST}_i(x, y)) \\
&\quad \vee \bigvee_{\substack{j, j' \\ j+j'=i+1}} (\text{LIST}_j(x, u) \wedge \text{LIST}_{j'}(u, y)) && \text{for } i \in \{1, \dots, \ell\} \\
\varphi_{\text{DEL}_U}^{\text{LAST}_i}(u; x) &\stackrel{\text{def}}{=} (\bigwedge_{i' \leq i} \neg \text{LAST}_{i'}(u) \wedge \text{LAST}_i(x)) \\
&\quad \vee \bigvee_{i' \leq i} (\text{LAST}_{i'}(u) \wedge \text{LIST}_{i-i'+1}(x, u)) && \text{for } i \in \{1, \dots, \ell\} \\
\varphi_{\text{DEL}_U}^{\text{FIRST}_i}(u; x) &\stackrel{\text{def}}{=} (\bigwedge_{i' \leq i} \neg \text{FIRST}_{i'}(u) \wedge \text{FIRST}_i(x)) \\
&\quad \vee \bigvee_{i' \leq i} (\text{FIRST}_{i'}(u) \wedge \text{LIST}_{i-i'+1}(u, x)) && \text{for } i \in \{1, \dots, \ell\}
\end{aligned}$$

$$\begin{aligned} \varphi_{\text{DELU}}^{\text{IS}_i}(u) &\stackrel{\text{def}}{=} \bigvee_{\substack{j,j' \\ j+j'-2=i}} (\text{FIRST}_j(u) \wedge \text{LAST}_{j'}(u)) && \text{for } i \in \{0, \dots, k\} \\ \varphi_{\text{DELU}}^{\text{IS}_{>k}}(u) &\stackrel{\text{def}}{=} \text{IS}_{>k} \wedge \bigwedge_{\substack{j,j' \\ j+j'-2=k}} (\neg \text{FIRST}_j(u) \vee \neg \text{LAST}_{j'}(u)) \end{aligned}$$

As all auxiliary relations can be maintained under insertions and deletions of elements, the presented dynamic program witnesses that SIZE- k is in binary DynProp. \square

Later on we will use that the dynamic program from Example 2.1 can straightforwardly be extended to maintain the set of nodes in a graph with degree k , for each fixed $k \in \mathbb{N}$: for each node v of the graph, the dynamic program maintains whether the set of nodes adjacent to v has size k . Towards this end, the program maintains one instance of the auxiliary relations from Example 2.1 for each node v . This is realised by increasing the arity of every auxiliary relation by one, and the additional dimension is used to indicate the node to which a tuple belongs. Accordingly, this dynamic program uses at most ternary auxiliary relations.

3. PARITY EXISTS AND QUANTIFIER-FREE UPDATES

In this section we start our examination of the PARITY EXISTS query in the context of quantifier-free update rules. Let us first formalise 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 PARITY EXISTS asks, given a coloured graph, whether the number of covered nodes is odd.

As stated in the introduction, PARITY EXISTS 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 PARITY EXISTS_{deg $\leq k$} of PARITY EXISTS asks whether the number of covered nodes that additionally have in-degree at most k is odd. Note that PARITY EXISTS_{deg $\leq k$} is a query on general coloured graphs, not only on graphs with bounded degree.

Theorem 1.2. PARITY EXISTS_{deg $\leq k$} can be maintained in DynProp with auxiliary relations of arity k , for any $k \geq 3$, but not with auxiliary relations of arity $k - 1$, even on graphs with in-degree at most k .

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

Theorem 1.1. PARITY EXISTS \notin DynProp.

Corollary 1.3. 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 PARITY EXISTS_{deg $\leq k$} which satisfies the conditions by Theorem 1.2.

³We 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 $(v, v) \in E$.

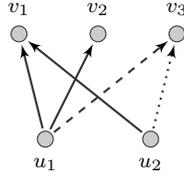


FIGURE 1. The graph used in the proof of Corollary 1.3. If the dashed edge is included, the graph is a positive instance of PARITYDEGREEDIV3 . If the dotted edge is included, it is a negative instance.

For $k = 2$, already [ZS15, 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 a dynamic program \mathcal{P} that only uses nullary auxiliary relations and consider an input instance that contains the nodes $V = \{u_1, u_2, v_1, v_2, v_3\}$ and the 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. See Figure 1 for an illustration. So, \mathcal{P} does not maintain PARITYDEGREEDIV3 . \square

The rest of this section is devoted to the proof of Theorem 1.2. First, in Subsection 3.1, we show that $\text{PARITYEXISTS}_{\text{deg} \leq k}$ can be maintained with k -ary auxiliary relations, for $k \geq 3$. For this, we employ the list technique introduced in Example 2.1. 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 [Zeu17].

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 3.1. *For every $k \geq 3$, $\text{PARITYEXISTS}_{\text{deg} \leq k}$ is in k -ary DynProp .*

Proof. Let $k \geq 3$ be some fixed natural number. We show how a DynProp -program \mathcal{P} can maintain $\text{PARITYEXISTS}_{\text{deg} \leq k}$ 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

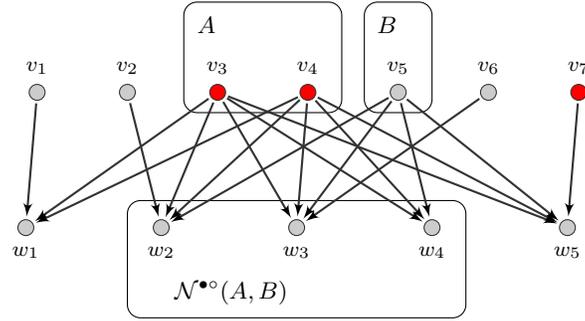


FIGURE 2. An illustration of the notation used in the proof of Proposition 3.1. The set $\mathcal{N}^{\bullet o}(A, B)$ does not include w_1 , as there is no edge (v_5, w_1) , and it does not include w_5 , as there is an edge (v_7, w_5) for a coloured node $v_7 \notin A$.

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 $c(v_1, \dots, v_k)$ 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, \dots, v_{k+1} .

We now present the construction in more detail. A node w is called *active* if its in-degree $\text{in-deg}(w)$ is at most k . Let $A = \{a_1, \dots, a_\ell\}$ be a set of coloured nodes and let $B = \{b_1, \dots, b_m\}$ be a set of uncoloured nodes, with $\ell + m \leq k$. By $\mathcal{N}_G^{\bullet o}(A, B)$ 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, $w \in \mathcal{N}_G^{\bullet o}(A, B)$ 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 $\mathcal{N}^{\bullet o}(A, B)$ if the graph G is clear from the context. See Figure 2 for an example. The dynamic program \mathcal{P} maintains the parity of $|\mathcal{N}_G^{\bullet o}(A, B)|$ 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_ℓ^\bullet with the intended meaning that for a node w it holds $w \in N_\ell$ if $\text{in-deg}(w) = \ell$ and $w \in N_\ell^\bullet$ if w has exactly ℓ coloured in-neighbours. These relations can be maintained as explained in Example 2.1 and the subsequent remark, requiring some additional, ternary auxiliary relations. We also use a relation $\text{ACTIVE} \stackrel{\text{def}}{=} N_1 \cup \dots \cup N_k$ 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 $(a_1, \dots, a_\ell, b_1, \dots, b_m)$ is contained in $P_{\ell, m}$ if and only if

- the nodes $a_1, \dots, a_\ell, b_1, \dots, b_m$ 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 $A = \{a_1, \dots, a_\ell\}$ and $B = \{b_1, \dots, b_m\}$.

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

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

Of course, \mathcal{P} also maintains the Boolean query relation ANS .

We now describe the update formulas of \mathcal{P} for the relations $P_{\ell, m}$ and ANS , assuming that each change actually alters the input graph, so, for example, no changes $\text{INS}_E(v, w)$ 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 $\text{INS}_R(v)$ 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 ANS is therefore

$$\varphi_{\text{INS}_R}^{\text{ANS}}(v) \stackrel{\text{def}}{=} \text{ANS} \oplus P_{0,1}(v).$$

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 $\bar{a} = a_1, \dots, a_\ell$ and $\bar{b} = b_1, \dots, b_m$ that $\mathcal{N}^{\bullet\circ}(\{a_1, \dots, a_\ell\}, \{b_1, \dots, b_m\})$ is of odd size after a change. The other update formulas are simple variants.

$$\begin{aligned}
\varphi_{\text{INS}_R}^{P_{\ell,m}}(v; x_1, \dots, x_\ell, y_1, \dots, y_m) &\stackrel{\text{def}}{=} \\
&\bigvee_{i \in [\ell]} (v = x_i \wedge P_{\ell-1,m+1}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_\ell, \bar{y}, v)) \\
&\vee \left(\bigwedge_{i \in [\ell]} v \neq x_i \wedge \bigwedge_{i \in [m]} v \neq y_i \wedge (P_{\ell,m}(\bar{x}, \bar{y}) \oplus P_{\ell,m+1}(\bar{x}, \bar{y}, v)) \right) \quad \text{for } \ell \geq 1, \ell + m < k \\
\varphi_{\text{INS}_R}^{P_{\ell,m}}(v; x_1, \dots, x_\ell, y_1, \dots, y_m) &\stackrel{\text{def}}{=} \\
&\bigvee_{i \in [\ell]} (v = x_i \wedge P_{\ell-1,m+1}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_\ell, \bar{y}, v)) \\
&\vee \left(\bigwedge_{i \in [\ell]} v \neq x_i \wedge \bigwedge_{i \in [m]} v \neq y_i \wedge P_{\ell,m}(\bar{x}, \bar{y}) \right) \quad \text{for } \ell \geq 1, \ell + m = k
\end{aligned}$$

Uncolouring a node v . The update formulas for a change $\text{DEL}_R(v)$ are analogous to the update formulas for a change $\text{INS}_R(v)$ as seen above. Again we only present a subset of the update formulas, the others are again easy variants.

$$\begin{aligned}
\varphi_{\text{DEL}_R}^{\text{ANS}}(v) &\stackrel{\text{def}}{=} \text{ANS} \oplus P_{1,0}(v) \\
\varphi_{\text{DEL}_R}^{P_{\ell,m}}(v; x_1, \dots, x_\ell, y_1, \dots, y_m) &\stackrel{\text{def}}{=} \\
&\bigvee_{i \in [m]} (v = y_i \wedge P_{\ell+1,m-1}(\bar{x}, v, y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_m)) \\
&\vee \left(\bigwedge_{i \in [\ell]} v \neq x_i \wedge \bigwedge_{i \in [m]} v \neq y_i \wedge (P_{\ell,m}(\bar{x}, \bar{y}) \oplus P_{\ell+1,m}(\bar{x}, v, \bar{y})) \right) \quad \text{for } m \geq 1, \ell + m < k \\
\varphi_{\text{DEL}_R}^{P_{\ell,m}}(v; x_1, \dots, x_\ell, y_1, \dots, y_m) &\stackrel{\text{def}}{=} \\
&\bigvee_{i \in [m]} (v = y_i \wedge P_{\ell+1,m-1}(\bar{x}, v, y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_m)) \\
&\vee \left(\bigwedge_{i \in [\ell]} v \neq x_i \wedge \bigwedge_{i \in [m]} v \neq y_i \wedge P_{\ell,m}(\bar{x}, \bar{y}) \right) \quad \text{for } m \geq 1, \ell + m = k
\end{aligned}$$

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

$$\varphi_{\text{INS}_E}^{\text{ANS}}(v, w) \stackrel{\text{def}}{=} \text{ANS} \oplus \left((N_k(w) \wedge \bigvee_{i \in [k]} N_i^\bullet(w)) \vee (R(v) \wedge N_0^\bullet(w) \wedge \bigvee_{i \in [k]} N_{i-1}(w)) \right).$$

The necessary updates for relations $P_{\ell, m}$ are conceptionally very similar. We list the conditions that characterise 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

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

- 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.

$$\begin{aligned} \psi_2 &\stackrel{\text{def}}{=} \bigvee_{i \in [\ell]} (v = x_i \wedge \bigwedge_{j \in [\ell] \setminus \{i\}} E(x_j, w) \wedge \bigwedge_{j \in [m]} E(y_j, w) \wedge N_{\ell-1}^\bullet(w) \wedge \bigvee_{j \in [k]} N_{j-1}(w)) \\ \psi_3 &\stackrel{\text{def}}{=} \bigvee_{i \in [m]} (v = y_i \wedge \bigwedge_{j \in [m] \setminus \{i\}} E(y_j, w) \wedge \bigwedge_{j \in [\ell]} E(x_j, w) \wedge N_\ell^\bullet(w) \wedge \bigvee_{j \in [k]} N_{j-1}(w)) \end{aligned}$$

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

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

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. The update formula for the query relation ANS is therefore

$$\begin{aligned} \varphi_{\text{DEL}_E}^{\text{ANS}}(v, w) &\stackrel{\text{def}}{=} \text{ANS} \oplus \left((N_{k+1}(w) \wedge \bigvee_{i \in [k+1]} N_i^\bullet(w) \wedge (\neg R(v) \vee \neg N_1^\bullet(w))) \right. \\ &\quad \left. \vee (R(v) \wedge N_1^\bullet(w) \wedge \bigvee_{i \in [k]} N_i(w)) \right) \end{aligned}$$

Regarding the update of relations $P_{\ell, m}$, we distinguish the same cases as for insertions $\text{INS}_E(v, w)$ 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. That means the active node w has incoming edges from all nodes from $A \cup B$, has no coloured in-neighbours apart from the nodes in A , and $v \in A \cup B$. This is expressed by the formula

$$\begin{aligned} \psi'_1 \stackrel{\text{def}}{=} & \bigwedge_{i \in [\ell]} E(x_i, w) \wedge N_\ell^\bullet(w) \wedge \bigwedge_{i \in [m]} E(y_i, w) \wedge \bigvee_{i \in [k]} N_i(w) \\ & \wedge \left(\bigvee_{i \in [\ell]} x_i = v \vee \bigvee_{i \in [m]} y_i = v \right). \end{aligned}$$

- Before the change, $w \in \mathcal{N}^{\bullet\circ}(A, B)$ does not hold, but it does afterwards. Then w already is and stays connected to all nodes from $A \cup B$, and either have degree $k + 1$ and become active and/or loose an additional coloured in-neighbour v . The following formula ψ'_2 lists the possible combinations.

$$\begin{aligned} \psi'_2 \stackrel{\text{def}}{=} & \bigwedge_{i \in [\ell]} (v \neq x_i \wedge E(x_i, w)) \wedge \bigwedge_{i \in [m]} (v \neq y_i \wedge E(y_i, w)) \\ & \wedge \left((N_{k+1}(w) \wedge \neg R(v) \wedge N_\ell^\bullet(w)) \right. \\ & \quad \vee (N_{k+1}(w) \wedge R(v) \wedge N_{\ell+1}^\bullet(w)) \\ & \quad \left. \vee \left(\bigvee_{i \in [k]} N_i(w) \wedge R(v) \wedge N_{\ell+1}^\bullet(w) \right) \right) \end{aligned}$$

Finally, the update formula for $P_{\ell, m}$ is

$$\varphi_{\text{DEL}_E}^{P_{\ell, m}}(v, w; x_1, \dots, x_\ell, y_1, \dots, y_m) \stackrel{\text{def}}{=} \theta_{\ell, m}(\bar{x}, \bar{y}) \wedge (P_{\ell, m}(\bar{x}, \bar{y}) \oplus (\psi'_1 \vee \psi'_2)). \quad \square$$

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 2.1 and the subsequent remark. In fact, this cannot be circumvented, as formalised by the next proposition.

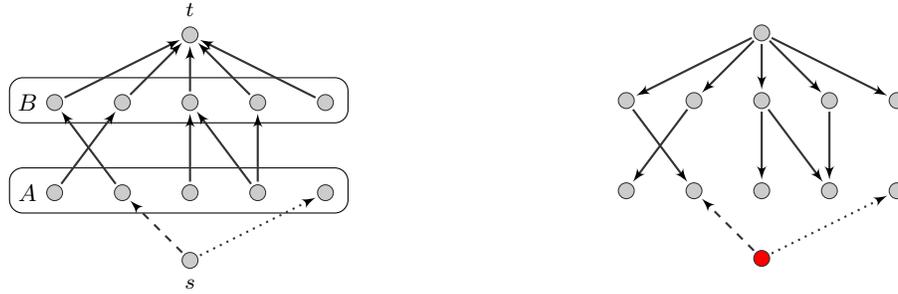
Proposition 3.2. *For $k \in \{1, 2\}$, $\text{PARITYEXISTS}_{\text{deg} \leq k}$ is not in binary DynProp, even with arbitrary initialisation.*

The proof relies on a lower bound result by Zeume and Schwentick [ZS15]. They show that one cannot maintain in binary DynProp, not even with arbitrary initialisation, whether there is a directed path from some distinguished node s to some distinguished node t in a 2-layered graph G [ZS15, Theorem 4.7]. A graph $G = (V, E, s, t)$ with distinguished nodes $s, t \in V$ is 2-layered, if its set V of nodes can be partitioned into sets $V = \{s, t\} \cup A \cup B$, such that edges go either from s to some node in A , from some node in A to some node in B , or from some node in B to t . So, the edge set E is a subset of $(\{s\} \times A) \cup (A \times B) \cup (B \times \{t\})$.

We prove Proposition 3.2 using a reduction from (a special case of) this query.

Proof. We first show the result for $k = 1$. The proof of [ZS15, Theorem 4.7] shows that there is no dynamic program with quantifier-free update rules and binary auxiliary relations that can maintain s - t -reachability in 2-layered graphs, not even if the auxiliary relations may be initialised arbitrarily. The proof actually shows that such dynamic programs cannot even maintain the query if

- the initial graph may be any 2-layered graph $G = (V, E, s, t)$ with some node set $V = \{s, t\} \cup A \cup B$ that satisfies the following conditions:
 - there is no edge from s to any other node, and



(A) A 2-layered graph G_u . If the dashed edge is included, the graph contains an s - t -path. If the dotted edge is included, the graph has no such path.

(B) A coloured graph G'_u . If the dashed edge is included, it is a negative instance of $\text{PARITYEXISTS}_{\text{deg} \leq 1}$. If the dotted edge is included, it is a positive instance.

FIGURE 3. An example for the construction in the proof of Proposition 3.2.

- all nodes from B have an edge to t ,
- the auxiliary relations are initialised arbitrarily,
- the changes consist of a sequence of deletions of edges from $A \times B$ followed by the insertion of a single edge from s to some node in A .

We now assume, towards a contradiction, that there is a dynamic program \mathcal{P}' that witnesses that $\text{PARITYEXISTS}_{\text{deg} \leq 1}$ is in binary DynProp. We show that from \mathcal{P}' we can construct a dynamic program \mathcal{P} with quantifier-free update rules and binary auxiliary relations that can maintain s - t -reachability for 2-layered graphs with the restrictions noted above, contradicting [ZS15, Theorem 4.7].

Let $G = (V, E, s, t)$ be some 2-layered graph with node set $V = \{s, t\} \cup A \cup B$, such that $(b, t) \in E$ for every $b \in B$ and $(s, v) \notin E$ for every $v \in V$. Let $G' = (V, E', R)$ be the coloured graph with the same node set V as G , edge set $E' \stackrel{\text{def}}{=} \{(v, u) \mid (u, v) \in E\}$, and $R = \emptyset$. Let \mathcal{A}' be the auxiliary relations that \mathcal{P}' assigns to G' starting from an initially empty graph and arbitrarily initialised auxiliary relations when the edges E' are inserted in some arbitrary order.

We now explain how a dynamic program \mathcal{P} can maintain s - t -reachability for G under deletions of edges from $A \times B$ followed by the insertion of a single edge from s to some node in A , starting from the initial auxiliary relations \mathcal{A}' . The basic idea is that changes to G are translated to changes to G' such that

- if the graph obtained from G has an s - t -path then the graph obtained from G' has no covered node with in-degree at most 1, and
- if the graph obtained from G has no s - t -path then the graph obtained from G' has exactly one covered node with in-degree at most 1.

The dynamic program \mathcal{P} proceeds as follows. If an edge $(u, v) \in E$ with $u \in A$ and $v \in B$ is deleted, then \mathcal{P} simulates \mathcal{P}' for the deletion of the edge $(v, u) \in E'$. If an edge (s, a) is inserted into E , then \mathcal{P} simulates \mathcal{P}' for the insertion of (s, a) into E' followed by the insertion of s into R . Then \mathcal{P} gives the query answer “true” if and only if the answer of \mathcal{P}' is “false”. All this is clearly expressible by quantifier-free update formulas.

Let G_u be the 2-layered graph that results from G by applying the changes, and let G'_u be the coloured graph that results from G' by applying the corresponding changes. An

example is depicted in Figure 3. It remains to show that there is an s - t -path in G_u if and only if the number of covered nodes in G'_u that have in-degree at most 1 is not odd.

Suppose that there is an s - t -path in G_u using edges $(s, a), (a, b), (b, t)$, for some nodes $a \in A$ and $b \in B$. It follows that the edges (s, a) and (b, a) are present in G'_u , so, a has in-degree at least 2. As a is the only neighbour of the only red node s , there are 0 covered nodes in G'_u with in-degree at most 1, an even number. Suppose on the other hand that there is no s - t -path in G_u . That means that the only node $a \in A$ such that an edge (s, a) exists in G_u has no edge to any node in B , because every node from B has an edge to t . Consequently, the only incoming edge of a in G'_u is from s , which is a red node. So, there is exactly one covered node in G'_u with in-degree at most 1, an odd number. This concludes the case $k = 1$.

The case of $k = 2$ can be proven along the same lines, with the only adjustment that in the graph G' we add edges from t to every node in $A \cup B$, increasing the in-degree of every node from this set by 1. \square

3.2. Inexpressibility results for $\text{PARITYEXISTS}_{\text{deg} \leq k}$. In this subsection we prove that k -ary auxiliary relations are not sufficient to maintain $\text{PARITYEXISTS}_{\text{deg} \leq k+1}$, for every $k \in \mathbb{N}$. The proof technique we use, and formalise as Lemma 3.3, is a reformulation of the proof technique of Zeume [Zeu17], which combines proof techniques from dynamic complexity [GMS12, ZS15] with insights regarding upper and lower bounds for Ramsey numbers. We actually use a special case of the formalisation from [SVZ18, 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 $\{1, \dots, n\}$ of elements that may be changed, for an arbitrary n , there is a structure \mathcal{I} whose domain includes the elements $\{1, \dots, n\}$ and a sequence $\delta_1(x_1), \dots, \delta_{k+1}(x_{k+1})$ of changes such that

- (1) the elements $1, \dots, n$ cannot be distinguished by quantifier-free formulas evaluated on \mathcal{I} , and
- (2) the structure that results from \mathcal{I} by applying the changes $\delta_1(i_1), \dots, \delta_{k+1}(i_{k+1})$ in that order, for each choice of $k + 1$ elements $\{i_1, \dots, i_{k+1}\} \subseteq \{1, \dots, n\}$, is a positive instance for q exactly if $\{i_1, \dots, i_{k+1}\} \in \mathcal{B}$.

In the following, we 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 3.3 [SVZ18]. *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 $\delta_1(x_1), \dots, \delta_{k+1}(x_{k+1})$ of changes

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

$$(\delta_1(p_{i_1}) \circ \dots \circ \delta_{k+1}(p_{i_{k+1}}))(\mathcal{I}) \in q \iff \{i_1, \dots, i_{k+1}\} \in \mathcal{B}.$$

With the help of Lemma 3.3 we can show the desired inexpressibility result.

Proposition 3.4. *For every $k \geq 0$, $\text{PARITYEXISTS}_{\text{deg} \leq k+1}$ is not in k -ary DynProp, even with arbitrary initialisation. This even holds if the input graph may only have in-degree at most $k+1$.*

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

Proof. Let $k \in \mathbb{N}$ be fixed. We apply Lemma 3.3 to show that $\text{PARITYEXISTS}_{\text{deg} \leq k+1}$ 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 $\mathcal{N}_{\exists}^{\rightarrow}(\{p_i \mid i \in B\})$ 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 $\text{PARITYEXISTS}_{\text{deg} \leq k+1}$ if and only if $B \in \mathcal{B}$. An invocation of Lemma 3.3 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 $\{(p_i, B) \mid B \in \mathcal{B}, i \in B\}$: 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 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 $|\mathcal{N}_{\exists}^{\rightarrow}(\{p_i \mid i \in B\})|$ 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 $p_{i_1}, \dots, p_{i_{k+1}}$ are marked, the number of covered nodes is odd precisely if there is one node in S that has an edge from *all* nodes $p_{i_1}, \dots, p_{i_{k+1}}$, which is the case exactly if $\{i_1, \dots, i_{k+1}\} \in \mathcal{B}$.

We now make this precise. Let n be arbitrary and let $P = \{p_1, \dots, p_n\}$. 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 $G = (V, E, R)$ with nodes $V \stackrel{\text{def}}{=} P \cup S$, where $S \stackrel{\text{def}}{=} \binom{[n]}{1} \cup \dots \cup \binom{[n]}{k+1}$, and initially empty set $R \stackrel{\text{def}}{=} \emptyset$ of coloured nodes. The edge set E contains all edges (p_i, Y) such that $i \in Y$ and the set $\{B \in \mathcal{B} \mid Y \subseteq B\}$ has odd size. See Figure 4 for an example of this construction. Note that for each $Y \in \binom{[n]}{i}$, for $i \in \{1, \dots, k+1\}$, the degree of Y in G is at most i , and therefore also at most $k+1$.

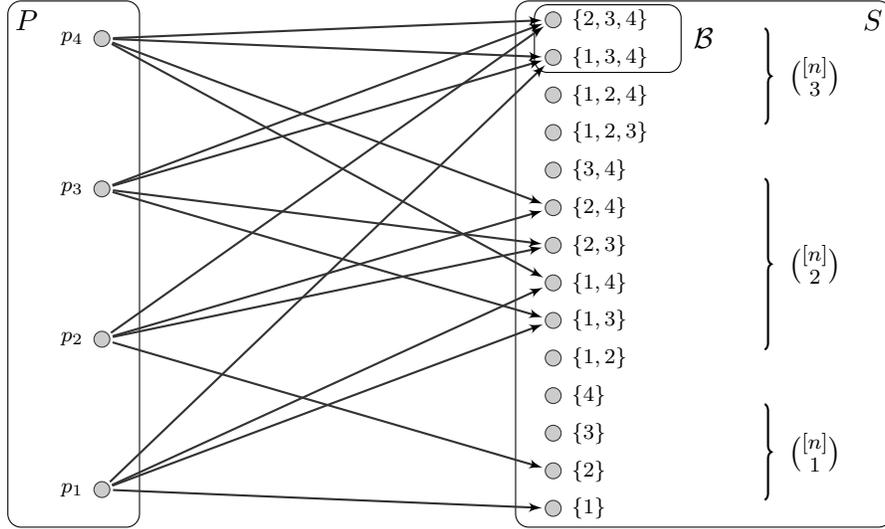


FIGURE 4. Example for the construction in the proof of Proposition 3.4, with $k = 2$ and $n = 4$.

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

$$|\mathcal{N}_{\exists}^{\rightarrow}(P_B)| = \sum_{i \in B} |\mathcal{N}_{\exists}^{\rightarrow}(p_i)| - \sum_{\substack{i, j \in B \\ i < j}} |\mathcal{N}_{\exists}^{\rightarrow}(\{p_i, p_j\})| + \cdots + (-1)^k |\mathcal{N}_{\exists}^{\rightarrow}(P_B)|.$$

For the parity of $|\mathcal{N}_{\exists}^{\rightarrow}(P_B)|$ it therefore holds

$$|\mathcal{N}_{\exists}^{\rightarrow}(P_B)| \equiv \sum_{\emptyset \subsetneq X \subseteq B} |\mathcal{N}_{\exists}^{\rightarrow}(P_X)| \pmod{2}.$$

Our claim follows from the observation that $|\mathcal{N}_{\exists}^{\rightarrow}(P_X)|$ is odd if and only if $X \in \mathcal{B}$, which we will prove next.

Recall that for each $i \leq n$ and $Y \in S$ an edge (p_i, Y) exists precisely if $i \in Y$ and Y is included in an odd number of sets $B' \in \mathcal{B}$. This means that $Y \in \mathcal{N}_{\exists}^{\rightarrow}(P_X)$ if and only if $X \subseteq Y$ and Y is included in an odd number of sets $B' \in \mathcal{B}$. The parity of $|\mathcal{N}_{\exists}^{\rightarrow}(P_X)|$ is therefore congruent modulo 2 to the number of pairs (Y, B') such that $X \subseteq Y \subseteq B'$ and $B' \in \mathcal{B}$ hold: if Y is included in an even number of sets $B' \in \mathcal{B}$ then $Y \notin \mathcal{N}_{\exists}^{\rightarrow}(P_X)$, but the parity is not affected by also counting the even number of pairs (Y, B') for this set Y .

For a fixed set B' , the number of sets Y with $X \subseteq Y \subseteq B'$ is $2^{|B' \setminus X|}$. We conclude that

$$|\mathcal{N}_{\exists}^{\rightarrow}(P_X)| \equiv \sum_{B' \in \mathcal{B}, X \subseteq B'} 2^{|B' \setminus X|} \pmod{2},$$

where the right-hand side is odd if and only if there is a $B' \in \mathcal{B}$ with $X \subseteq B'$ and $B' \setminus X = \emptyset$, so if $X \in \mathcal{B}$. This proves the claim.

Let $\delta_1(x_1), \dots, \delta_{k+1}(x_{k+1})$ be the change sequence $\text{INS}_R(x_1), \dots, \text{INS}_R(x_{k+1})$ that colours the nodes x_1, \dots, x_{k+1} . Let $B \in \binom{[n]}{k+1}$ be of the form $\{i_1, \dots, i_{k+1}\}$ with $i_1 < \dots < i_{k+1}$. The change sequence $\alpha_B \stackrel{\text{def}}{=} \delta_1(p_{i_1}) \cdots \delta_{k+1}(p_{i_{k+1}}) = \text{INS}_R(p_{i_1}), \dots, \text{INS}_R(p_{i_{k+1}})$ results in a graph where the set of coloured nodes is exactly P_B . As all nodes in $\mathcal{N}_{\exists}^{\rightarrow}(P_B)$ have degree at most $k+1$ and the set $\mathcal{N}_{\exists}^{\rightarrow}(P_B)$ is of odd size exactly if $B \in \mathcal{B}$, we have that $\alpha_B(\mathcal{I})$ is a positive instance of $\text{PARITYEXISTS}_{\text{deg} \leq k+1}$ if and only if $B \in \mathcal{B}$. \square

4. PARITYEXISTS AND FIRST-ORDER UPDATES

As discussed in the introduction, the PARITY query can be easily maintained with first-order update rules. So far we have seen that its generalisation 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 full 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 trivial⁴.

We start by providing a dynamic program with first-order update rules that maintains $\text{PARITYEXISTS}_{\text{deg} \leq k}$, 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 $\text{PARITYEXISTS}_{\text{deg} \leq \log n}$ with binary auxiliary relations.

Theorem 1.5. *$\text{PARITYEXISTS}_{\text{deg} \leq k}$ 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 for $\text{PARITYEXISTS}_{\text{deg} \leq k}$ need auxiliary relations of growing arity 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, \dots , 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 $I \subseteq \{1, \dots, k\}$ that *indexes* the neighbours. With this idea, the proof of Proposition 3.1 can be adjusted appropriately for Theorem 1.5.

Proof (of Theorem 1.5). 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.

⁴If the out-degree of a node is upper-bounded by some constant c and the colour of some node v changes, then first-order update rules can determine the number $i \leq c$ of nodes that have an incoming edge from v and no coloured in-neighbour $w \neq v$. The change of the query result is determined by the parity of i .

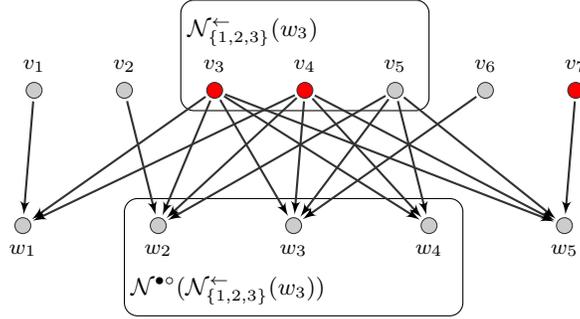


FIGURE 5. An illustration of the notation used in the proof of Theorem 1.5. The set $\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{\{1,2,3\}}^{\leftarrow}(w_3))$ does not include w_1 , as there is no edge (v_5, w_1) , and it does not include w_5 , as there is an edge (v_7, w_5) for a coloured node $v_7 \notin \mathcal{N}_{\{1,2,3\}}^{\leftarrow}(w_3)$. Compare with Figure 2.

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 $\mathcal{N}_I^{\leftarrow}(w)$ 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 3.1. For a graph G and an arbitrary set C of (coloured and uncoloured) nodes, we denote by $\mathcal{N}_G^{\bullet\circ}(C)$ 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 . An example for these notions is depicted in Figure 5.

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 $\mathcal{N}_I^{\leftarrow}(w)$, and
- (2) the set $\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))$ has odd size.

Note that (1) implies that $w \in \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))$.

An auxiliary relation P_I basically replaces the relations $P_{\ell, m}$ with $\ell + m = |I|$ from the proof of Proposition 3.1, and the updates are mostly analogous.

We now explain how the query relation ANS and the relations P_I are updated when a modification to the input graph occurs. For maintaining P_I , observe that first-order formulas can easily express that a node w is active, has at least $\max(I)$ neighbours and satisfies condition (1). The main task of the dynamic program we construct is to check condition (2).

Colouring a node v . When a change $\text{INS}_R(v)$ occurs, so, when a node v is coloured, the query relation ANS 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 is an active and previously uncovered node w that v has an edge to and, furthermore, 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.

For an arbitrary index set I , 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 let G' be the changed graph. Let w be any active node. If v is an I -indexed in-neighbour of w , so if $v \in \mathcal{N}_I^{\leftarrow}(w)$, no change regarding $w \in P_I$ is necessary. Otherwise if $v \notin \mathcal{N}_I^{\leftarrow}(w)$, we need to check whether

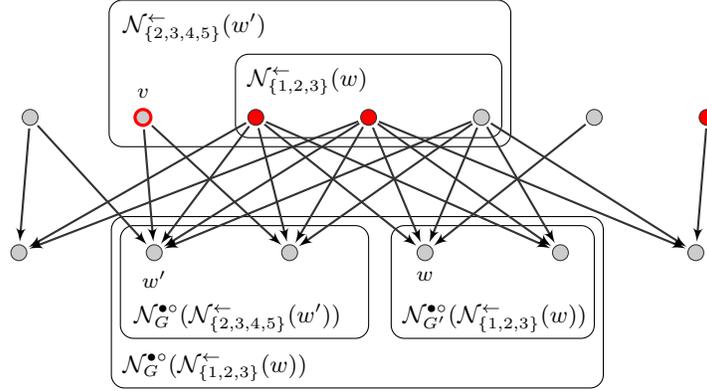


FIGURE 6. Illustration for the update of the relation P_I in the proof of Theorem 1.5, for $I = \{1, 2, 3\}$. Graph G' results from graph G by colouring node v . The set $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_{1,2,3}^{\leftarrow}(w))$ is the disjoint union of the sets $\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{2,3,4,5}^{\leftarrow}(w'))$ and $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_{1,2,3}^{\leftarrow}(w))$, so the parity of $|\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_{1,2,3}^{\leftarrow}(w))|$ can be determined from the parity of $|\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{2,3,4,5}^{\leftarrow}(w'))|$ and the parity of $|\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{1,2,3}^{\leftarrow}(w))|$.

both conditions (1) and (2) from above are satisfied regarding w and I . If at least one of the conditions is not satisfied, then w is not contained in the updated relation P_I .

Suppose that condition (1) is satisfied, so, that w has no coloured in-neighbours in G' that are not contained in $\mathcal{N}_I^{\leftarrow}(w)$. This can easily be checked by a first-order formula. We need to check condition (2), so, whether $|\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))|$ is odd.

The set $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))$ contains exactly those nodes from $\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))$ that do not have the newly coloured node v as in-neighbour, as this node is not contained in $\mathcal{N}_I^{\leftarrow}(w)$. So, $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w)) = \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w)) \setminus \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w) \cup \{v\})$. It follows that we can determine the parity of $|\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))|$ from the parity of $|\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))|$ and the parity of $|\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w) \cup \{v\})|$. We know whether $|\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))|$ is odd, this is the case if and only if $w \in P_I$. It remains to find the parity of $|\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w) \cup \{v\})|$.

Suppose that $\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w) \cup \{v\})$ is non-empty and contains some node w' . The node w' is active and has an edge from v and every node in $\mathcal{N}_I^{\leftarrow}(w)$, but no edge from any other coloured node. Such a node w' can easily be identified by a first-order formula. Let I' be chosen such that $\mathcal{N}_{I'}^{\leftarrow}(w') = \mathcal{N}_I^{\leftarrow}(w) \cup \{v\}$. The set $\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w) \cup \{v\})$ coincides with the set $\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{I'}^{\leftarrow}(w'))$ and the size of this set is odd exactly if $w' \in P_{I'}$.

So, in this case, condition (2) is satisfied in G' exactly if $P_I(w) \oplus P_{I'}(w')$ holds for the old auxiliary relations. An illustration of the reasoning is given in Figure 6.

Note that if no such w' exists, then $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w)) = \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))$ and no change regarding $w \in P_I$ is necessary.

Uncolouring a node v . The update of the query relation ANS and the relations P_I after a change $\text{DEL}_R(v)$ that uncolours a node v are symmetric to the previous case. The query bit is flipped if there is an active node w that has v as its only coloured in-neighbour and if $w \in P_I$ for the index set $I = \{i\}$, where i is the number such that v is the i -th in-neighbour of w . The update of auxiliary relations P_I for some node w is exactly as for the previous case of changes $\text{INS}_R(v)$ — for checking condition (2) in the case $v \notin \mathcal{N}_I^{\leftarrow}(w)$, we now have that $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))$ is the disjoint union of $\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(w))$ and $\mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{I'}^{\leftarrow}(w'))$, where I' and w'

are defined exactly as in the previous case, still condition (2) holds exactly if $P_I(w) \oplus P_{I'}(w')$ holds.

Inserting an edge (v, w) . When a change $\text{INS}_E(v, w)$ inserts an edge (v, w) , the query bit ANS needs to be flipped if w is a covered node that becomes inactive, so if its degree becomes larger than k , or if w stays active and v becomes its first coloured in-neighbour. Otherwise, ANS is not changed.

For any index set I , the relation P_I is updated as follows. Let G be the graph before the change, and let G' be the changed graph that includes the edge (v, w) . First, we consider whether some active node $u \neq w$ is in the updated version of P_I . We assume that condition (1) from above holds for u , otherwise $u \notin P_I$ before and after the update. Whether $u \in P_I$ holds might only change if the parity of $\mathcal{N}_I^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(u))$ changes. This only happens when the membership of w in $\mathcal{N}_I^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(u))$ changes by inserting (v, w) , and this happens if

- (a) $w \in \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(u))$ and
 - (i) w becomes inactive, or
 - (ii) v is a coloured node that is not included in $\mathcal{N}_I^{\leftarrow}(u)$; or
- (b) $w \notin \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(u))$ and w stays active, the node v is in $\mathcal{N}_I^{\leftarrow}(u)$, and w has an incoming edge from all nodes from $\mathcal{N}_I^{\leftarrow}(u)$ in G' and no incoming edge from any coloured node that is not in this set.

It remains to explain how to determine whether $w \in P_I$ needs to hold after the update. In the following, we make the graph explicit in the notation $\mathcal{N}_I^{\leftarrow}(w)$ and write $\mathcal{N}_{G,I}^{\leftarrow}(w)$ for the set if I -indexed in-neighbours of w in G . We assume that w stays active and that condition (1) from above holds after the change, so w has no coloured in-neighbours that are not in $\mathcal{N}_{G',I}^{\leftarrow}(w)$. The update formulas only need to check whether condition (2) holds, that is, whether the set $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_{G',I}^{\leftarrow}(w))$ has odd size.

We consider two cases. First, suppose $v \notin \mathcal{N}_{G',I}^{\leftarrow}(w)$. As w satisfies condition (1), it follows that v is uncoloured. Let I' be the index set such that $\mathcal{N}_{G',I}^{\leftarrow}(w) = \mathcal{N}_{G,I'}^{\leftarrow}(w)$. Then $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_{G',I}^{\leftarrow}(w)) = \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{G,I'}^{\leftarrow}(w))$, and $w \in P_I$ holds after the update if and only if $w \in P_{I'}$ before the update.

Now, suppose $v \in \mathcal{N}_{G',I}^{\leftarrow}(w)$. If w is not the only node in $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_{G',I}^{\leftarrow}(w))$, then there is another active node w' and an index set I' such that $\mathcal{N}_{G',I}^{\leftarrow}(w) = \mathcal{N}_{G',I'}^{\leftarrow}(w')$ and w' has no coloured in-neighbour apart from nodes in $\mathcal{N}_{G',I'}^{\leftarrow}(w')$. It follows that $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_{G',I}^{\leftarrow}(w)) = \{w\} \cup \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{G',I'}^{\leftarrow}(w'))$. So, $w \in P_I$ holds after the update if and only if $w' \notin P_{I'}$ before the update.

All conditions can easily be expressed by first-order formulas.

Deleting an edge (v, w) . The update after a change $\text{DEL}_E(v, w)$ is mostly symmetric to the update after the insertion of an edge (u, v) . We shortly describe it here. The query bit ANS needs to be flipped if w is a covered and formerly inactive node that becomes active, so, if w has degree k after applying the change and has a coloured in-neighbour different from v . The bit ANS also needs to be flipped if w is a covered and active node before applying the change, but v is the only node that covers w . In all other cases, ANS is not changed.

We now discuss the update of P_I , for some index set I . Let G be the graph before the change, and let G' be the changed graph that does not include the edge (v, w) . For a node $u \neq w$, the reasoning for updating whether $u \in P_I$ holds is very similar to the reasoning in

the case of inserting an edge (u, v) . The only differences are the conditions that tell whether the membership of w in $\mathcal{N}^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(u))$ changes. This happens after the edge (v, w) is deleted, if

- (a) $w \in \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(u))$ and the node v is in $\mathcal{N}_I^{\leftarrow}(u)$, or
- (b) $w \notin \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_I^{\leftarrow}(u))$, the node w is active in G' , has an incoming edge from all nodes from $\mathcal{N}_I^{\leftarrow}(u)$ in G' , but not from any coloured node that is not in this set, and
 - (i) w was not active in G , or
 - (ii) v is a coloured node that is not included in $\mathcal{N}_I^{\leftarrow}(u)$.

Now we explain the update of P_I for the node w . We assume that w is active in G' and satisfies condition (1) from above; otherwise, $w \notin P_I$. We need to explain how to determine the parity of $|\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_{G',I}^{\leftarrow}(w))|$. We again consider two cases. First, suppose that w was already active in G and that v is uncoloured. Let I' be the index set such that $\mathcal{N}_{G',I}^{\leftarrow}(w) = \mathcal{N}_{G',I'}^{\leftarrow}(w)$. Again, $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_{G',I}^{\leftarrow}(w)) = \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{G',I'}^{\leftarrow}(w))$, and $w \in P_I$ holds after the update if and only if $w \in P_{I'}$ before the update.

Suppose the first case does not apply. Then, if w is not the only node in $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_{G',I}^{\leftarrow}(w))$, there is again an active node $w' \neq w$ and an index set I' such that $\mathcal{N}_{G',I}^{\leftarrow}(w) = \mathcal{N}_{G',I'}^{\leftarrow}(w')$ and w' has no coloured in-neighbour apart from nodes in $\mathcal{N}_{G',I'}^{\leftarrow}(w')$. It holds that $\mathcal{N}_{G'}^{\bullet\circ}(\mathcal{N}_{G',I}^{\leftarrow}(w)) = \{w\} \cup \mathcal{N}_G^{\bullet\circ}(\mathcal{N}_{G',I'}^{\leftarrow}(w'))$ and so, $w \in P_I$ after the update if and only if $w' \notin P_{I'}$ before the update.

Again, all conditions are easily seen to be first-order expressible. \square

It is easy to maintain a linear order on the non-isolated nodes of an input graph [Ete98], which is all that is needed for the proof of Theorem 1.5. So, $\text{PARITYEXISTS}_{\text{deg} \leq k}$ 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 the proof of Theorem 1.5 for $\text{PARITYEXISTS}_{\text{deg} \leq k}$ to PARITYEXISTS , but only to $\text{PARITYEXISTS}_{\text{deg} \leq \log n}$, 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 1.4. *$\text{PARITYEXISTS}_{\text{deg} \leq \log n}$ 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 $\{0, \dots, n-1\}$, and use BIT to access the bit encoding of these numbers. Any node $v \in V$ then naturally encodes a set $I(v) \subseteq \{1, \dots, \log n\}$: $i \in \{1, \dots, \log n\}$ 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 1.5 constructs a dynamic program that maintains unary relations P_I , for each non-empty set $I \subseteq \{1, \dots, k\}$. We replace these relations by a single binary relation P , with the intended meaning that $(v, w) \in P$ if and only if $w \in P_{I(v)}$. First-order update rules can easily translate between these two representations in the presence of a linear order and BIT, and otherwise the update works exactly as described in the proof of Theorem 1.5. \square

In addition to a linear order, Etessami [Ete98] 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. [Imm99, 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 $\text{PARITYEXISTS}_{\text{deg} \leq \log n}$ 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. MAINTENANCE USING AUXILIARY RELATIONS OF QUASI-POLYNOMIAL SIZE

Orthogonally to the perspectives taken in this work so far, which focussed on the expressive power of the update formalism, 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. The class DynFO allows for polynomially many auxiliary bits: the auxiliary relations of a DynFO program can be encoded by a bit string of polynomial size. It is not hard to see that if one allows quasi-polynomially many auxiliary bits – so, the number of auxiliary bits is bounded by $2^{\log^{O(1)} n}$ – 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. Here, we provide a proof sketch for this observation.

For discussing the amount of auxiliary bits, it is convenient to switch the view point from first-order updates to updates computed by AC^0 circuits. A classical result linking circuit complexity and finite model theory states that a query can be computed by a uniform family of AC^0 -circuits (that is, by constant depth and polynomial size circuits with \neg , \wedge - and \vee -gates with unbounded fan-in) if and only if it can be expressed by a first-order formula with access to a built-in linear order and BIT [BIS90]. So, if we assume the presence of a built-in linear order and BIT then the classes DynFO and (uniform) DynAC^0 coincide.

The class ACC^0 is defined similarly as AC^0 , but the circuits are additionally allowed to use modulo-gates. A query can be computed by a uniform family of ACC^0 -circuits if and only if it can be expressed by a first-order formula that may use modulo quantifiers, in addition to a linear order and BIT.

For simplifying the discussion, in the following we take a solely circuit-based perspective. We also, from now on, disregard uniformity conditions and only consider non-uniform circuit classes.

The classes q-AC^0 and q-ACC^0 are defined as the classes AC^0 and ACC^0 except that circuits can be of quasi-polynomial size, that is, of size $2^{\log^{O(1)} n}$. The class DynAC^0 is the class of queries that can be maintained with AC^0 -circuits and polynomially many auxiliary bits. The class q-DynAC^0 is defined as the class DynAC^0 except that dynamic programs may use quasi-polynomially many auxiliary bits and update circuits from q-AC^0 .

It turns out that with quasi-polynomial update circuits all q-ACC^0 -queries can be maintained, and in particular the query PARITYEXISTS .

Theorem 5.1. *Every query in q-ACC^0 can be maintained in q-DynAC^0 .*

Instead of proving this theorem directly, we use that q-ACC^0 can be characterised by very simple circuits with one gate with quasi-polynomial in-degree [BT94].

A boolean function $f : \{0, 1\}^m \rightarrow \{0, 1\}$ is *symmetric* if $f(\bar{x}) = f(\bar{y})$ whenever the number of ones in \bar{x} and \bar{y} is equal. The class Sym^+ contains all queries computable by depth-two size- $2^{\log^{O(1)} n}$ circuits where the output gate computes a symmetric boolean function, and it has incoming wires from a layer of and-gates, which each have fan-in $\log^{O(1)} n$ (see Reference [BT94]).

As $q\text{-ACC}^0$ is contained in Sym^+ [BT94, Proposition 1.2], the following result implies Theorem 5.1.

Theorem 5.2. *Every query in Sym^+ can be maintained in $q\text{-DynAC}^0$.*

Proof. The proof extends the ideas from the proofs of Theorem 1.4 and Theorem 1.5 to nodes with in-degree $\log^{\mathcal{O}(1)} n$.

Let \mathcal{C} be a family of Sym^+ -circuits, where for each $n \in \mathbb{N}$ the circuit C_n is a depth-two size- $2^{\log^{\mathcal{O}(1)} n}$ circuit whose output gate computes some symmetric boolean function h , and that otherwise consists of a layer of and-gates g_1, g_2, \dots with fan-in at most k , for some $k \in \log^{\mathcal{O}(1)} n$.

We construct a $q\text{-DynAC}^0$ -program \mathcal{P} . The idea is to maintain, for each domain size n , the number m of and-gates of C_n that are currently *activated*, i.e., that are only connected to inputs that are currently set to 1. The output of the symmetric function h for inputs with m ones can then be looked up in a table.

For maintaining the number of activated and-gates, the program \mathcal{P} maintains, for every subset A of the input gates of size at most k , the value

$$\#(A) \stackrel{\text{def}}{=} |\{g_i \mid g_i \text{ is connected to all inputs in } A, \\ \text{and is activated when the inputs from } A \text{ are ignored}\}|,$$

that is, the number of and-gates that are connected to all inputs in A and whose only inputs that are not set to 1 (if any) are contained in A .

Rephrased in the setting of the previous sections, we can think of the circuit as a coloured graph G , and an input gate is considered to be coloured if the input bit is set to 0. Then, $\#(A)$ is the number of and-gates that are connected to all inputs in A and are *not* covered by some node that is *not* in A , so, the cardinality of the set $\mathcal{N}_G^\bullet(A)$ as defined in Section 4.

As the number of sets of input gates of size at most k is quasi-polynomial in n , and for each such set A the number $\#(A)$ is bounded by the size of the circuit, which is again quasi-polynomial in n , this auxiliary information can be encoded by quasi-polynomially many auxiliary bits.

We need to show that the number of activated gates as well as the auxiliary information $\#(A)$ can be updated after changes that flip an input bit. Note that the circuit itself is fixed, so the dynamic program does not need to support changes that delete or insert wires.

When an input bit x is changed, the values $\#(A)$ are updated to $\#'(A)$ as follows. For all sets A that include x , the count $\#(A)$ does not change, as all inputs in A are ignored in the definition of $\#(A)$. Also, the count $\#(A)$ does not change for sets A with $x \notin A$ and $|A| = k$, as all and-gates have fan-in at most k and can therefore not be connected to all inputs in A and the additional input x . The only case left is for a set A with $|A| < k$ that does not include x . Suppose that x is flipped from 0 to 1. All and-gates counted for $\#(A)$ are not connected to x by the definition of $\#(A)$. After the change, also those gates are counted for $\#(A)$ that have a connection to all gates in A and to x , but to no other input gate that is set to 0. The number of these gates is given by $\#(A \cup \{x\})$.

In summary, the updated auxiliary information $\#'(A)$ after x is set from 0 to 1 is as follows:

$$\#'(A) \stackrel{\text{def}}{=} \begin{cases} \#(A) & \text{if } x \in A \text{ or } |A| = k \\ \#(A) + \#(A \cup \{x\}) & \text{else.} \end{cases}$$

If x is set from 1 to 0, the update is symmetric:

$$\#'(A) \stackrel{\text{def}}{=} \begin{cases} \#(A) & \text{if } x \in A \text{ or } |A| = k \\ \#(A) - \#(A \cup \{x\}) & \text{else.} \end{cases}$$

The total number of activated gates changes by the number $\#(\{x\})$, which is either added in case x is flipped from 0 to 1, or subtracted if x is flipped from 1 to 0. These updates can easily be expressed by q-AC⁰ circuits. \square

In Reference [Muk19] it is discussed how Theorem 5.1 can be extended to show that all queries in q-AC can be maintained in q-DynAC⁰, using different techniques. Here, q-AC denotes the class of all queries that can be computed by families of circuits with quasi-polynomial size and poly-logarithmic depth.

6. 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 PARITYEXISTS_{deg ≤ log n} is in DynFO and that PARITYEXISTS_{deg ≤ 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?*

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 682588.

REFERENCES

- [AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.
- [Ajt83] Miklós Ajtai. Σ_1^1 -formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.
- [BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC¹. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990.
- [BT94] Richard Beigel and Jun Tarui. On ACC. *Computational Complexity*, 4:350–366, 1994.
- [DKM⁺18] 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.
- [DMS⁺19] 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*, 15(2), 2019.
- [DS98] 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.
- [DZ00] 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.

- [Ete98] Kousha Etessami. Dynamic tree isomorphism via first-order updates. In *PODS*, pages 235–243, 1998.
- [FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.
- [GMS12] Wouter Gelade, Marcel Marquardt, and Thomas Schwentick. The dynamic complexity of formal languages. *ACM Trans. Comput. Log.*, 13(3):19, 2012.
- [Hes03] William Hesse. *Dynamic Computational Complexity*. PhD thesis, University of Massachusetts Amherst, 2003.
- [Imm99] Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.
- [Muk19] Anish Mukherjee. *Static and Dynamic Complexity of Reachability, Matching and Related Problems*. PhD thesis, Chennai Mathematical Institute, 2019.
- [PI97] Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel, dynamic complexity class. *J. Comput. Syst. Sci.*, 55(2):199–209, 1997.
- [SVZ18] Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. Dynamic complexity under definable changes. *ACM Trans. Database Syst.*, 43(3):12:1–12:38, 2018.
- [SZ16] Thomas Schwentick and Thomas Zeume. Dynamic complexity: recent updates. *SIGLOG News*, 3(2):30–52, 2016.
- [Zeu17] Thomas Zeume. The dynamic descriptive complexity of k-clique. *Inf. Comput.*, 256:9–22, 2017.
- [ZS15] Thomas Zeume and Thomas Schwentick. On the quantifier-free dynamic complexity of reachability. *Inf. Comput.*, 240:108–129, 2015.