Dynamic Complexity of Reachability: How Many Changes Can We Handle?

Samir Datta
Chennai Mathematical Institute, India
sdatta@cmi.ac.in

Pankaj Kumar
Chennai Mathematical Institute, India
Department of Applied Mathematics, Charles University, Prague, Czech Republic
pankaj@kam.mff.cuni.cz

Anish Mukherjee
Institute of Informatics, University of Warsaw, Poland
anish@mimuw.edu.pl

Anuj Tawari
Chennai Mathematical Institute, India
atawari@cmi.ac.in

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

Thomas Zeume
Ruhr-Universität Bochum, Germany
thomas.zeume@rub.de

Abstract

In 2015, it was shown that reachability for arbitrary directed graphs can be updated by first-order formulas after inserting or deleting single edges. Later, in 2018, this was extended for changes of size $\frac{\log n}{\log \log n}$, where n is the size of the graph. Changes of polylogarithmic size can be handled when also majority quantifiers may be used.

In this paper we extend these results by showing that, for changes of polylogarithmic size, first-order update formulas suffice for maintaining (1) undirected reachability, and (2) directed reachability under insertions. For classes of directed graphs for which efficient parallel algorithms can compute non-zero circulation weights, reachability can be maintained with update formulas that may use "modulo 2" quantifiers under changes of polylogarithmic size. Examples for these classes include the class of planar graphs and graphs with bounded treewidth. The latter is shown here.

As the logics we consider cannot maintain reachability under changes of larger sizes, our results are optimal with respect to the size of the changes.

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

Keywords and phrases Dynamic complexity, reachability, complex changes

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.122

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version A full version of this paper is available at `http://arxiv.org/abs/2004.12739`.

Funding *Samir Datta*: The author was partially funded by a grant from Infosys foundation and SERB-MATRICS grant MTR/2017/000480.
Pankaj Kumar: The author was supported by Czech Science Foundation GAČR (grant #19-27871X).
Anish Mukherjee: The author was supported by the ERC CoG grant TUgbOAT no 772346.
Nils Vortmeier: The author acknowledges the financial support by DFG grant SCHW 678/6-2.

Acknowledgements The first author would like to thank Chetan Gupta for finding a problem in a previous version of the proof of Theorem 14.

1 Introduction

Suppose we are given a graph G whose edge relation is subjected to insertions and deletions of edges. Which resources are required to update the reachability relation of the graph?

Recently it was shown that if one is allowed to store auxiliary relations, then the reachability relation can be updated after single edge insertions and deletions using first-order logic formulas with access to the graph, the stored relations, and the changed edges [4]. In other words, the reachability query is contained in the dynamic complexity class DynFO [19]. From a database perspective, this means that it can be updated with core-SQL queries; from the perspective of circuit complexity, this means that reachability can be updated by circuits of polynomial size in constant-time due to the correspondence of first-order logic and AC^0 established by Barrington, Immerman, and Straubing [1].

Understanding single edge insertions and deletions is an important first step. Yet in applications, changes to a graph G often come as bulk set ΔE of changed edges. It is natural to ask, how large the set ΔE of edges can be such that reachability can be maintained with the same resources as for single edge changes – that is with first-order formulas or, respectively, AC^0 circuits. Using existing lower bounds for circuits [21], it is easy to see that DynFO (or DynAC^0, respectively) cannot handle changes of size larger than polylogarithmic for many queries, including the reachability query (see Section 3 for a more detailed discussion).

The best one can hope for is to maintain reachability with first-order formulas for changes of polylogarithmic size with respect to the size of the graph. In a first step, a subset of the authors showed that reachability can be maintained in $\mathsf{DynFO}(\leq,+,\times)$ under changes of size $O(\frac{\log n}{\log\log n})$ [7]. Here, the class $\mathsf{DynFO}(\leq,+,\times)$ extends DynFO by access to built-in arithmetic, which for technical reasons is more natural for bulk changes. Unfortunately, the techniques used in [7] seem only be able to handle changes of polylogarithmic size in the extension of DynFO by majority quantifiers, that is, in the class $\mathsf{DynFO}{+}\mathsf{Maj}(\leq,+,\times)$.

In this paper we make progress on handling changes of polylogarithmic size in DynFO by attacking the challenge from two directions. First, we establish two restrictions for which reachability can be maintained under these changes.

▶ **Main Theorem 1.** *Reachability can be maintained in* $\mathsf{DynFO}(\leq,+,\times)$ *under*
- *insertions of polylogarithmically many edges; and*
- *insertions and deletions of polylogarithmically many edges if the graph remains undirected.*

As second contribution of this paper, we provide a meta-theorem for establishing classes of graphs for which reachability can be maintained under polylogarithmic-size changes with a slight extension of first-order logic. In this extension, $\mathsf{DynFO}{+}\mathsf{Mod}\,2(\leq,+,\times)$, formulas used for updating the reachability information and the auxiliary relations may use parity quantifiers in addition to the traditional universal and existential quantifiers.

▶ **Main Theorem 2.** *Reachability can be maintained in* $\mathsf{DynFO}{+}\mathsf{Mod}\,2(\leq,+,\times)$ *under insertions and deletions of polylogarithmically many edges on classes of graphs for which polynomially bounded non-zero circulation weights can be computed in* AC.

Here a weighting function for the edges of a graph has *non-zero circulation*, if the weight of every directed cycle is non-zero (see Section 6 for details). The class AC contains queries computable by circuits of polynomial size and polylogarithmic depth. Examples

for graph classes for which non-zero circulation weights can be computed in AC include the class of planar graphs and graphs with bounded treewidth. The latter is shown here. We note that isolating weights, a concept closely related to non-zero circulation weights, have been used previously in dynamic complexity for establishing that reachability is in non-uniform DynFO+Mod $2(\leq, +, \times)$ under single edge changes [3], a precursor result to reachability in DynFO.

For our results, we employ two techniques of independent interest. The first technique relies on the power of first-order logic on structures of polylogarithmic size. It is well-known that reachability can be computed by a uniform circuit family of size $N^{\mathcal{O}(N^{1/d})}$ and depth $2d$. An immediate consequence is that all NL-queries can be expressed by first-order formulas for graphs with n nodes but only polylogarithmically many edges. Thus, for maintaining a query under changes ΔE of polylogarithmic size, a dynamic program can (1) do an arbitrary NL-computation on ΔE, and (2) update the auxiliary data by combining the computed information with the previous auxiliary data using a first-order formula.

The second technique we rely on is a slight generalization of the "Muddling Lemma" from [6]. The Muddling Lemma reduces the requirements for proving that a query is in DynFO: a query is in DynFO if, essentially, one can update the query for polylog many steps starting from auxiliary data precomputed in AC. Here we observe that this can be strengthened for changes of polylogarithmic size: a query is in DynFO if, essentially, one can update the query under one polylogarithmic-size change from auxiliary data precomputed in AC.

Parts of the results presented here have been included in the PhD thesis of Nils Vortmeier [24].

Outline. After recalling the dynamic complexity framework in Section 2, we shortly outline barriers for the size of bulk changes in Section 3 and recall useful techniques in Section 4. Afterwards we present our results for DynFO in Section 5 and for DynFO+Mod $2(\leq, +, \times)$ in Section 6.

2 The dynamic setting

We briefly repeat the essentials of dynamic complexity, closely following [7] which in turn builds on [20]. The goal of a *dynamic program* is to answer a given query on a relational *input structure* subjected to changes that insert tuples into the input relations or delete tuples from them. The program may use auxiliary information represented by an *auxiliary structure* over the same domain as the input structure. Initially, both input and auxiliary structure are empty; and the domain is fixed during each run of the program. Whenever a change to the input structure occurs, the auxiliary structure is updated by means of first-order formulas.

Changes. For a (relational) structure $\mathcal{I}$ over domain D and schema σ, a change $\Delta\mathcal{I}$ consists of sets R^+ and R^- of tuples for each relation symbol $R \in \sigma$. The result $\mathcal{I} + \Delta\mathcal{I}$ of an application of the change $\Delta\mathcal{I}$ to $\mathcal{I}$ is the input structure where $R^{\mathcal{I}}$ is changed to $(R^{\mathcal{I}} \cup R^+) \setminus R^-$. The *size* of $\Delta\mathcal{I}$ is the total number of tuples in relations R^+ and R^- and the set of *affected elements* is the (active) domain of tuples in $\Delta\mathcal{I}$.

Dynamic Programs and Maintenance of Queries. A dynamic program consists of a set of update rules that specify how auxiliary relations are updated after changing the input structure. Let $\mathcal{I}$ be the current input structure over schema σ and let $\mathcal{A}$ be the auxiliary

structure over some schema σ_{aux}. An *update rule* for updating an ℓ-ary auxiliary relation T after a change is a first-order formula φ over schema $\sigma \cup \sigma_{\text{aux}}$ with ℓ free variables. After a change $\Delta\mathcal{I}$, the new version of T is $T \stackrel{\text{def}}{=} \{\bar{a} \mid (\mathcal{I} + \Delta\mathcal{I}, \mathcal{A}) \models \varphi(\bar{a})\}$, so, the updated auxiliary relation includes all tuples $\bar{a}$ such that $\varphi(\bar{a})$ is satisfied when it is evaluated on the changed input structure and the old auxiliary structure. Note that a dynamic program can choose to have access also to the old input structure by storing it in its auxiliary relations.

For a state $\mathcal{S} = (\mathcal{I}, \mathcal{A})$ of the dynamic program $\mathcal{P}$ with input structure $\mathcal{I}$ and auxiliary structure $\mathcal{A}$ we denote the state of the program after applying a change sequence α and updating the auxiliary relations accordingly by $\mathcal{P}_\alpha(\mathcal{S})$.

The dynamic program *maintains* a q-ary query Q under changes of size k if it has a q-ary auxiliary relation Ans that at any time stores the result of Q applied to the current input structure. More precisely, for each non-empty sequence α of changes of size k, the relation Ans in $\mathcal{P}_\alpha(\mathcal{S}_\emptyset)$ and $Q(\alpha(\mathcal{I}_\emptyset))$ coincide, where the state $\mathcal{S}_\emptyset \stackrel{\text{def}}{=} (\mathcal{I}_\emptyset, \mathcal{A}_\emptyset)$ consists of an input structure $\mathcal{I}_\emptyset$ and an auxiliary structure $\mathcal{A}_\emptyset$ over some common domain that both have empty relations, and $\alpha(\mathcal{I}_\emptyset)$ is the input structure after applying α.

If a dynamic program maintains a query, we say that the query is in DynFO. Similarly to DynFO one can define variants with built-in auxiliary relations and with more powerful update formulas. For instance, the class DynFO$(\leq, +, \times)$ contains queries that can be maintained by first-order update formulas with access to three particular auxiliary relations $<, +$, and $\times$ which are initialized as a linear order and the corresponding addition and multiplication relations; in the class DynFO+Mod p, update formulas may use modulo-p-quantifiers in addition to existential and universal quantifiers.

We state our results for dynamic classes with access to the arithmetic relations $\leq, +$ and $\times$. Handling bulk changes without access to arithmetic leads to technical issues which distract from the fundamental dynamic properties. See [7, 24] for further discussions on this topic and how our results can be stated for DynFO in an adapted setting which takes these technical issues into account.

For the construction of dynamic programs in this paper we assume that changes either only insert edges or only delete edges. This is no restriction, as corresponding update formulas can be combined to process a change that inserts and deletes edges at the same time, by first processing the inserted edges and then processing the deleted edges.

3 Barriers for the size of bulk changes

In the following we outline why it is not possible to maintain reachability under changes of larger than polylogarithmic size with first-order formulas, even in the presence of parity quantifiers.

The idea is simple. A classical result by Smolensky states that for computing the number of ones modulo a prime q occurring in a bit string of length n, an AC$[p]$ circuit of depth d requires $2^{\Omega(n^{1/2d})}$ many gates, for each prime p distinct from q (see [21] or, for a modern exposition, [17, Theorem 12.27]). A simple, well-known reduction yields that deciding reachability for graphs with n edges which are disjoint unions of paths also requires AC$[p]$ circuits of size $2^{\Omega(n^{1/2d})}$. Indeed, computing the number of ones in $w = a_1 \cdots a_n$ modulo q can be reduced to reachability as follows. Consider the graph with nodes $\{(i,k) \mid 1 \leq i \leq n+1 \text{ and } 0 \leq k < q\}$ and edges $\{((i,k),(i+1,k)) \mid a_i = 0\} \cup \{((i,k),(i+1,k+1 \bmod q)) \mid a_i = 1\}$. It is easy to see that (i) the graph has $\mathcal{O}(n)$ edges and is a disjoint union of q paths, and (ii) there is a path from $(1,0)$ to $(n+1,0)$ if and only if the number of ones in w is 0 modulo q.

These lower bounds for circuit sizes immediately translate into lower bounds for first-order formulas with modulo p quantifiers via the correspondence due to Barrington, Immerman, and Straubing [1].

▶ **Theorem 1.** *Let $f(n) \in \log^{\omega(1)} n$ be a function from $\mathbb{N}$ to $\mathbb{N}$ and let p be a prime. There is no* FO+Mod p *formula with access to built-in relations that defines*

1. *whether the size of a unary relation U with $|U| \leq f(n)$ is divisible by q, for primes q distinct from p;*
2. *reachability in graphs with at most $f(n)$ edges, even for disjoint unions of paths.*

Proof sketch.

(a) Let $f(n)$ be some function from $\log^{\omega(1)} n$. Suppose, towards a contradiction, that there is an FO+Mod p formula with access to built-in relations that defines whether the size of a unary relation U with $|U| \leq f(n)$ is divisible by q, for some primes $p \neq q$. Then, by [1], for every n there is an AC[p] circuit of some fixed depth d that decides that question for inputs if size n, and the size of this circuit is polynomial in n. That is a contradiction, as by Smolensky's lower bound every such circuit needs to have size $2^{\Omega(f(n)^{1/2d})} = 2^{\log(n)^{\omega(1)}} = n^{\omega(1)}$.

(b) This part can be proven analogously to Part (a), using the circuit lower bound for graph reachability. ◀

Those lower bounds have the immediate consequence that DynFO cannot deal with bulk changes of larger than polylogarithmic size. Indeed, from any formula that updates the result of a query after an insertion of $f(n)$ tuples into an initially empty input relation one can construct a formula that defines the query for inputs of size $f(n)$.

▶ **Corollary 2.** *Let $f(n) \in \log^{\omega(1)} n$ be a function from $\mathbb{N}$ to $\mathbb{N}$ and let p be a prime. Then the following queries cannot be maintained in* DynFO+Mod p *for bulk changes of size $\leq f(n)$, even if the auxiliary relations may be initialized arbitrarily:*

1. *divisibility of the size of a unary relation by a prime $q \neq p$, and*
2. *reachability in graphs, even if restricted to disjoint unions of paths.*

4 Techniques and Tools

In the previous section we recalled that FO($\leq, +, \times$), even if equipped with modulo p quantifiers, is not very expressive in general. That changes when we are only interested in small substructures of our input: FO($\leq, +, \times$) can express every NL-computable query on subgraphs of polylogarithmic size.

▶ **Theorem 3.** *Let k and c be arbitrary natural numbers, and let Q be a k-ary,* NL*-computable graph query. There is an* FO($\leq, +, \times$) *formula φ over schema $\{E, D\}$ such that for any graph G with n nodes, any subset D of its nodes of size at most $\log^c n$ and any k-tuple $\bar{a} \in D^k$: $\bar{a} \in Q(G[D])$ if and only if $(G, D) \models \varphi(\bar{a})$. Here, $G[D]$ denotes the subgraph of G induced by D.*

Proof. We prove the result for the reachability query. As reachability is NL-complete under FO($\leq, +, \times$)-reductions [16], and every FO($\leq, +, \times$)-reduction maps an instance of size $\log^c n$ to an instance of size $\log^{cd} n$ for a fixed $d \in \mathbb{N}$, the full result follows.

It is well-known (see for example [2, p. 613]), that for every $d \in \mathbb{N}$ there is a uniform circuit family for reachability where the circuit for inputs of size N has depth $2d$ and size $N^{\mathcal{O}(N^{1/d})}$. Suppose the input size N is only $\log^c n$, for some $c \in \mathbb{N}$ and pick $d \stackrel{\text{def}}{=} 2c$. Then the circuit size

$$\begin{aligned} N^{\mathcal{O}(N^{1/d})} =& (\log^c n)^{\mathcal{O}((\log^c n)^{1/d})} \\ =& (\log n)^{c\mathcal{O}((\log n)^{c/d})} = (\log n)^{\mathcal{O}((\log n)^{c/2c})} = (\log n)^{\mathcal{O}(\sqrt{\log n})} \\ =& 2^{\mathcal{O}(\log\log n\sqrt{\log n})} \subseteq 2^{\mathcal{O}(\sqrt{\log n}\sqrt{\log n})} = 2^{\mathcal{O}(\log n)} = n^{\mathcal{O}(1)} \end{aligned}$$

is polynomial in n, so, the circuit is a uniform AC^0 circuit for reachability for graphs of size $\log^c n$. The existence of φ follows by the equivalence of uniform AC^0 and $\mathsf{FO}(\leq, +, \times)$ [1]. ◀

The Muddling Lemma simplifies the maintenance of queries under single edge changes [6]. It states that for many natural queries Q, in order to show that Q can be maintained, it is enough to show that the query can be maintained for a bounded number of steps. In the following we recall the necessary notions and extend the lemma to bulk changes.

A query Q is *almost domain-independent* if there is a $c \in \mathbb{N}$ such that $Q(\mathcal{A})[(\text{adom}(\mathcal{A}) \cup B)] = Q(\mathcal{A}[(\text{adom}(\mathcal{A}) \cup B)])$ for all structures $\mathcal{A}$ and sets $B \subseteq A \setminus \text{adom}(\mathcal{A})$ with $|B| \geq c$. Here, $\text{adom}(\mathcal{A})$ denotes the *active domain*, i.e. the set of domain elements that are used in some tuple of $\mathcal{A}$. A query Q is $(\mathcal{C}, f)$-*maintainable*, for some complexity class $\mathcal{C}$ and some function $f : \mathbb{N} \to \mathbb{R}$, if there is a dynamic program $\mathcal{P}$ and a $\mathcal{C}$-algorithm $\mathbb{A}$ such that for each input structure $\mathcal{I}$ over a domain of size n, each linear order $\leq$ on the domain, and each change sequence α of length $|\alpha| \leq f(n)$, the relation Q in $\mathcal{P}_\alpha(\mathcal{S})$ and $Q(\alpha(\mathcal{I}))$ coincide, where $\mathcal{S} = (\mathcal{I}, \mathbb{A}(\mathcal{I}, \leq))$.

The Muddling Lemma from [6] has been formulated for bulk changes in [7, 24][1].

▶ **Theorem 4** ([7, 24]). *Let Q be an* NL*-computable, almost domain independent query, and let $c \in \mathbb{N}$ be arbitrary. If the query Q is $(\mathsf{AC}^d, \log^d n)$-maintainable under changes of size $\log^c n$ for some $d \in \mathbb{N}$, then Q is in $\mathsf{DynFO}(\leq, +, \times)$ under changes of size $\log^c n$.*

The previous theorem can be strengthened as follows.

▶ **Theorem 5.** *Let Q be an* NL*-computable, almost domain independent query, and let $c \in \mathbb{N}$ be arbitrary. If the query Q is $(\mathsf{AC}^d, 1)$-maintainable under changes of size $\log^{c+d} n$ for some $d \in \mathbb{N}$, then Q is in $\mathsf{DynFO}(\leq, +, \times)$ under changes of size $\log^c n$.*

Proof. Let Q and d be as in the theorem statement, and let $\mathbb{A}$ be an AC^d algorithm and $\mathcal{P}$ a dynamic program that witness that Q is $(\mathsf{AC}^d, 1)$-maintainable under changes of size $\log^{c+d} n$. By Theorem 4 it suffices to show that there is an AC^d algorithm $\mathbb{A}'$ and a dynamic program $\mathcal{P}'$ that witness that Q is $(\mathsf{AC}^d, \log^d n)$-maintainable under changes of size $\log^c n$.

We choose $\mathbb{A}'$ as $\mathbb{A}$. The program $\mathcal{P}'$ just stores the at most $\log^{c+d} n$ changes that accumulate during the $\log^d n$ steps, and in each step uses $\mathcal{P}$ to answer Q, using the initial auxiliary relations computed by $\mathbb{A}$. ◀

5 Handling Polylog Changes with DynFO

So far we do not know how to maintain directed reachability under polylogarithmically many changes in $\mathsf{DynFO}(\leq, +, \times)$. In this section we show that reachability can be maintained in $\mathsf{DynFO}(\leq, +, \times)$ under insertions of polylogarithmically many edges for arbitrary graphs (disallowing any deletions), and under insertions *and* deletions of polylogarithmic size for undirected graphs.

[1] The statement and proof in [7] is slightly flawed and has been corrected in [24].

The general idea is similar in both cases. After changing polylogarithmically many edges with an effect on nodes V_{aff}, the dynamic program (1) computes a structure of polylogarithmic size on V_{aff}, (2) uses Theorem 3 to compute helpful information for this structure, and (3) updates the auxiliary relations by combining this information with the previous auxiliary data. Both (1) and (3) are performed by first-order formulas, and (2) uses an NL-computation.

▶ **Theorem 6.** *Reachability is in* DynFO$(\leq, +, \times)$ *under insertions of size* $\log^c n$*, for every* $c \in \mathbb{N}$*.*

Proof. Let $c \in \mathbb{N}$ be fixed. We construct a dynamic program with a single auxiliary relation Ans which stores the transitive closure of the current graph.

Whenever a set E^+ of edges is inserted into the current graph $G = (V, E)$, the dynamic program updates Ans with the help of the transitive closure relation of a graph H defined as follows. The nodes of H are the nodes V_{aff} affected by the change, that is, the nodes incident to edges in E^+. The edge set E_H of H contains the newly inserted edges E^+, and additionally edges (u, v) for all pairs (u, v) of nodes from V_{aff} that are connected by a path in G. Observe that H is of size $O(\log^c n)$ and first-order definable from G, E^+ and Ans. Hence, by Theorem 3, the transitive closure of H can be defined by a first-order formula.

The transitive closure relation of $G' \stackrel{\text{def}}{=} (V, E \cup E^+)$ can now be constructed from the transitive closures of G and H. To this end observe that the transitive closure of H equals the transitive closure relation of G' restricted to V_{aff}: it accounts for all paths from a node $u \in V_{\text{aff}}$ to another node $v \in V_{\text{aff}}$ that may use both newly inserted edges and edges that are already present in G. For this reason, every path ρ in G' consists of three consecutive subpaths $\rho_1\rho_2\rho_3 = \rho$, where ρ_1 and ρ_3 are defined as the maximal subpaths of ρ that do not rely on edges from E^+. These subpaths already exist in G and are represented in Ans. The subgraph ρ_2 by definition starts and ends at nodes from V_{aff}, so its existence is given by the transitive closure relation of H.

Hence, the transitive closure of G' can be defined by the formula $\varphi(s,t) \stackrel{\text{def}}{=} \textsc{Ans}(s,t) \vee \exists x_1 \exists x_2 \left(\textsc{Ans}(s,x_1) \wedge \mathrm{TC}_H(x_1,x_2) \wedge \textsc{Ans}(x_2,t)\right)$. ◀

▶ **Theorem 7.** *Reachability on undirected graphs can be maintained in* DynFO$(\leq, +, \times)$ *under changes of size* $\log^c n$*, for every* $c \in \mathbb{N}$*.*

Proof. The dynamic program from [9] that maintains undirected reachability in DynFO under single-edge changes uses, in addition to the transitive closure relation of the input graph, two binary auxiliary relations that represent a directed spanning forest of the input graph and its transitive closure, respectively. We show that these relations can still be maintained in DynFO$(\leq, +, \times)$ under changes of $\log^c n$ many edges, for fixed $c \in \mathbb{N}$.

Recall that it suffices to treat insertions and deletions independently, as they can be handled subsequently by a dynamic program.

For edge insertions, the construction idea is very similar to the proof of Theorem 6. We define a graph H, where nodes correspond to connected components of the input graph that include an affected node, and edges indicate that some inserted edge connects the respective connected components. As this graph is of polylogarithmic size, thanks to Theorem 3 we can express a spanning forest for H and its transitive closure in FO$(\leq, +, \times)$, which is sufficient to update the respective relations for the whole input graph.

In the case of edge deletions, the update formulas need to replace deleted spanning tree edges, whenever this is possible. Our approach is very similar to the case of edge insertions. The spanning tree decomposes into polylogarithmically many connected components when edges are deleted. These components can be merged again if non-tree edges exist that connect

them, and these edges become tree edges of the spanning forest. For a correspondingly defined graph of polylogarithmic size we can again define a spanning forest and its transitive closure, and from this information select the new tree edges.

We explain both cases in more detail. Let $G = (V, E)$ be the undirected input graph of size n with transitive closure Ans, and let S and TC_S be a directed spanning forest for G and its transitive closure, respectively. Suppose that a set E^+ of size at most $\log^c n$ is inserted. We define a graph H as follows. It contains a node $v \in V$ if (1) v is affected, that is, if E^+ contains an edge of v, and (2) v is the smallest affected node in its connected component of G with respect to $\leq$. It contains an edge (u, v) if $(u', v') \in E^+$ for some nodes u', v' with $(u, u') \in \text{Ans}$ and $(v, v') \in \text{Ans}$, so, if the connected components of u and v are connected by an inserted edge. The graph H is easily seen to be FO-definable using Ans. Because H is of polylogarithmic size with respect to n and a spanning forest of a graph can be computed[2] in NL, we can define a spanning tree S_H as well as its transitive closure TC_{S_H} in $\mathsf{FO}(\leq, +, \times)$, thanks to Theorem 3.

The update formulas define updated auxiliary relations for the graph $G' = (V, E \cup E^+)$ as follows. Intuitively, an edge $(u, v) \in S_H$ means that the connected components of u and v in G shall be connected in G' directly by a new tree edge. There might be several edges in E^+ that may serve this purpose, and we need to choose one of them. So, an edge $(u', v') \in E^+$ becomes part the updated spanning forest if there is an edge $(u, v) \in S_H$ such that u' and u as well as v' and v are in the same connected component of G, respectively, and (u', v') is the lexicographically minimal edge with these properties. This is clearly $\mathsf{FO}(\leq, +, \times)$-expressible using the old auxiliary relations. The old tree edges from S are taken over to the updated version, although some directions need to be inverted, if for a newly chosen tree edge (u', v') the node v' was not the root of the directed spanning tree of its connected component. First-order formulas that determine which edges need to be reversed and that provide the adjusted transitive closure for the components of the spanning forest are given in [9]. The relation TC_S is updated by combining this information with TC_{S_H}.

We note that Ans is first-order expressible from TC_S. In conclusion, all auxiliary relations can be updated in $\mathsf{FO}(\leq, +, \times)$.

Now suppose that a set E^- of at most $\log^c n$ edges is deleted. Let S' be the spanning forest that results from S after all tree edges from E^- are removed, and let $\mathrm{TC}_{S'}$ be its transitive closure, which is easily FO-expressible from TC_S. Similarly as above we define a graph H, with nodes being the minimal affected nodes in a weakly connected component of S', which are connected by an edge if the respective weakly connected components of S' are connected by some edge from $E \setminus E^-$. The same way as above, $\mathsf{FO}(\leq, +, \times)$ formulas can define a spanning forest and its transitive closure for H and then use this information to define a spanning forest and its transitive closure for the changed graph $G' = (V, E \setminus E^-)$. ◀

6 Handling Polylog Changes with DynFO+Mod 2

While we have seen, in the last section, that reachability for directed graphs can be maintained under edge insertions of polylogarithmic size, a matching result for edge deletions is still missing. Two intermediate results were shown in [7], building on the work of [13]: reachability can be maintained in $\mathsf{DynFO}(\leq, +, \times)$ under insertions and deletions that affect $\frac{\log n}{\log \log n}$ nodes, and in DynFO+Maj under insertions and deletions of polylogarithmically many edges.

[2] For example the breadth-first spanning forest with the minimal nodes of each component, with respect to $\leq$, as roots can be computed with the inductive counting technique due to Immerman and Szelepcsényi [15, 22].

In this section, we adapt the proof of the latter result, also using ideas that appear in [3], and show that reachability can be maintained in DynFO+Mod $2(\leq, +, \times)$ under edge insertions and deletions of polylogarithmic size for classes of directed graphs for which non-vanishing weight assignments can be computed in AC, that is, by polynomial-size circuits of polylogarithmic depth. This is possible for example for planar graphs [23], as well as for graphs with bounded treewidth, as we show towards the end of this section.

We start by giving the necessary definitions regarding isolating and non-vanishing weight assignments.

6.1 Isolating and non-vanishing weights

A *weighted* directed graph (G, w) consists of a graph $G = (V, E)$ and a *weight assignment* $w\colon E \to \mathbb{Z}$ that assigns an integer weight $w(e)$ to each edge $e \in E$. The weight assignment w is *bounded* by a function $f(|V|)$ if w assigns only weights from the interval $[-f(|V|), f(|V|)]$.

The weighted graph (G, w) is *min-unique* if (1) w only gives positive weights to the edges E, and (2) if some path from s to t exists, for some pair s, t of nodes, then there is a unique path from s to t with minimum weight under w. Here, the weight of a path (and in general every sequence of edges) is the sum of the weights of its edges. If (G, w) is min-unique, we say that w *isolates* (minimal paths in) G.

Define $\vec{G} = (V, \vec{E})$ to be the *bidirected extension* of G, where $\vec{E} \stackrel{\text{def}}{=} \{(u, v), (v, u) \mid (u, v) \in E\}$. A weight assignment w is *skew-symmetric* if $w(u, v) = -w(v, u)$ for all $(u, v) \in \vec{E}$. It has *non-zero circulation* if the weight of every simple directed cycle in $\vec{G}$ is non-zero (here, a cycle is *simple* if no node occurs twice).

From polynomially bounded non-zero circulation weights for $\vec{G}$ we can easily compute isolating weights for G.

▶ **Lemma 8.** *Let $G = (V, E)$ be a graph with n nodes, and let w be skew-symmetric non-zero circulation weight assignment for $\vec{G}$, which is bounded by n^k for some $k \in \mathbb{N}$. Then w' with $w'(e) = w(e) + n^{k+2}$ for every $e \in E$ isolates G.*

Proof. All weights in w' are clearly positive. It remains to show that w' isolates minimal paths in G. Assume, towards a contradiction, that there are two different s-t-paths ρ_1, ρ_2 with the same minimal weight under w' in G, for some nodes s and t. Without loss of generality, they are both simple paths, as otherwise they cannot be minimal. Let u be the last node visited by both paths before they differ for the first time, and let v be the first node after u that is visited by both paths. Let ρ_1^{uv}, ρ_2^{uv} be the subpaths in ρ_1, ρ_2 from u to v, respectively. If these subpaths have different weights, say, $w'(\rho_1^{uv}) < w'(\rho_2^{uv})$, then we can replace ρ_2^{uv} by ρ_1^{uv} in ρ_2 and get a lighter path, contradicting the assumption that both ρ_1 and ρ_2 are paths with minimal weight. So, $w'(\rho_1^{uv}) = w'(\rho_2^{uv})$ needs to hold. Then also $w(\rho_1^{uv}) = w(\rho_2^{uv})$ holds, because $w(\rho)$ and $w'(\rho)$ differ by a multiple of n^{k+2} for any path ρ, and the difference between $w(\rho)$ and $w(\rho')$ is at most n^{k+1}, for simple paths ρ and ρ'. So, w' cannot compensate weight differences under w. But then the concatenation of ρ_1^{uv} and the reverse of ρ_2^{uv} is a simple cycle in $\vec{G}$ with weight $w(\rho_1^{uv}) - w(\rho_2^{uv}) = 0$, contradiction the assumption that w has non-zero circulation. ◀

We explain how (families of) polynomially bounded weight assignments for graphs are represented in relational structures. Let V be the node set of a weighted graph of size n. We identify V with the set $\{0, \dots, n-1\}$ of numbers according to the given linear order $\leq$. A tuple $(a_1, \dots, a_k)$ of nodes then represents the number $\sum_{i=1}^{k} a_i n^{i-1}$. A (partial) function $f\colon V^k \to V^\ell$ is represented as a $(k+\ell)$-ary relation F over V, such that for each $\bar{a} \in V^k$ there

is at most one $\bar{b} \in V^\ell$ with $(\bar{a}, \bar{b}) \in F$. We say that f is $\mathsf{FO}(\leq, +, \times)$-definable if F is defined by an $\mathsf{FO}(\leq, +, \times)$ formula $\psi(\bar{x}, \bar{y})$, where $\bar{x} = x_1, \ldots, x_k$ and $\bar{y} = y_1, \ldots, y_\ell$. An $\mathsf{FO}(\leq, +, \times)$ formula $\psi(\bar{z}, \bar{x}, \bar{y})$, with $\bar{z} = z_1, \ldots, z_m$, defines a family $\{f(\bar{c})\colon V^k \to V^\ell \mid \bar{c} \in V^m\}$ of functions.

6.2 Maintaining Reachability in weighted graphs

We now state and prove the main result of this section.

▶ **Theorem 9.** *Let $\mathcal{G}$ be a class of graphs for which polynomially bounded skew-symmetric non-zero circulation weights can be computed in* AC*. Then, reachability for graphs in $\mathcal{G}$ is in* $\mathsf{DynFO{+}Mod}\,2(\leq, +, \times)$ *under changes of size* $\log^c n$*, for every $c \in \mathbb{N}$.*

Although this result leaves open whether reachability can be maintained in $\mathsf{DynFO}(\leq, +, \times)$ under polylogarithmically many edge changes, note that, in light of Corollary 2, it gives a tight upper bound for the size of changes that can be handled in $\mathsf{DynFO{+}Mod}\,2(\leq, +, \times)$.

We outline the proof strategy, which closely follows the strategy from [7]. Suppose, we are given a weighted directed graph (G, w) where $G = (V, E)$ is a graph with n nodes and w is an isolating weight assignment. We represent this weighted graph by an $n \times n$ matrix $A_{(G,w)}(x)$ as follows: if $(u, v) \in E$, then the u-v-entry of $A_{(G,w)}(x)$ is $x^{w(u,v)}$, where x is a formal variable, otherwise the u-v-entry is 0.

The matrix $D \stackrel{\text{def}}{=} \sum_{i=0}^{\infty}(A_{G_w}(x))^i$ is a matrix of formal power series in the formal variable x, and from an s-t-entry $\sum_{i=0}^{\infty} c_i x^i$ of this matrix we can read the number c_i of paths from s to t with weight i. Our goal is to determine the coefficients c_i modulo 2, for all i up to some polynomial bound. From this information we can deduce whether there is a path from s to t in G: as w isolates minimal paths in G, if there is some path from s to t, then there is a unique path with minimal weight, which means that for the weight ℓ of this path we have $c_\ell \equiv 1 \pmod 2$. Otherwise, if no path from s to t exists, $c_\ell \equiv 0 \pmod 2$ for all i.

We use the following insights to actually compute and update the coefficients c_i. Notice that the matrix D is invertible over the ring of formal power series (see [7] and its full version [8]) and can be written as $D = (I - A_{(G,w)}(x))^{-1}$, where I is the identity matrix.

So, we need to compute and update the inverse of a matrix. This cannot be done effectively for matrices of inherently infinite formal power series. For this reason we compute D only approximately. A *b-approximation* C of D, for some $b \in \mathbb{N}$, is a matrix of formal polynomials that agrees with the entries of D on the low-degree coefficients c_i for all $i \leq b$. This precision is preserved by the matrix operations we use, see [7, Proposition 14]. Note that it is sufficient to maintain an approximation of D, as for a weighted graph with polynomially bounded weights the maximal possible weight $w_{\max}$ of a minimal path is bounded by a polynomial, and thus only the coefficients c_i with $i \leq w_{\max}$ are relevant.

To update the matrix inverse, we employ the Sherman-Morrison-Woodbury identity (cf. [12]). This identity states that when updating a matrix A to a matrix $A + \Delta A$, with ΔA writeable as matrix product UBV, the inverse of A can be updated as follows:

$$(A + \Delta A)^{-1} = (A + UBV)^{-1} = A^{-1} - A^{-1}U(I + BVA^{-1}U)^{-1}BVA^{-1}.$$

When ΔA has only k non-zero rows and columns, there is a decomposition UBV where B is a $k \times k$ matrix.

The right-hand side can be computed in $\mathsf{FO{+}Mod}\,2(\leq, +, \times)$ for $k \stackrel{\text{def}}{=} \log^c n$. To see this, we observe that also $I + BVA^{-1}U$ is a $k \times k$ matrix. Computing the right-hand side now requires multiplication and iterated addition of polynomials over $\mathbb{Z}$ as well as the computation

of the inverse of a $k \times k$ matrix. As all computations are done modulo 2, this is indeed possible in $\mathsf{FO}{+}\mathsf{Mod}\,2(\leq, +, \times)$ for (matrices of) polynomials with polynomial degree using results of [11]. We provide more details later.

As we work with isolating weight assignments, our update routines also need to assign weights to changed edges such that the resulting weight assignment is again isolating. We show that this can be done if we start with (slightly adjusted) non-zero circulation weights. Using Theorem 5 we can assume that such an assignment is given, and that we only need to update the weights once.

Proof sketch (of Theorem 9). Let c be arbitrary. Thanks to Theorem 5 it suffices to show that there is a $d \in \mathbb{N}$ such that reachability is $(\mathsf{AC}^d, 1)$-maintainable by a dynamic program $\mathcal{P}$ under changes of size $\log^{c+d} n$. Let $d' \in \mathbb{N}$ be such that polynomially bounded skew-symmetric non-zero circulation weights for graphs from $\mathcal{G}$ can be computed in $\mathsf{AC}^{d'}$, and set $d \stackrel{\text{def}}{=} \max(2, d')$.

Let $G = (V, E)$ be a graph with n nodes. Let u be skew-symmetric non-zero circulation weights for G and let n^k be the polynomial bound on the weights. Further, let w be the weight assignment that gives weight $n^{k+2} + u(e)$ to each edge $e \in E$. Notice that w is polynomially bounded by n^{k+3} and isolates G according to Lemma 8.

The AC^d initialization computes, as auxiliary information, the weightings u and w and an n^b-approximation C of $(I - A_{(G,w)}(x))^{-1} \bmod 2$, that is, a matrix of formal polynomials in x that agree with the formal power series in $(I - A_{(G,w)}(x))^{-1} \bmod 2$ on the coefficients up to degree n^b. Here, $b \in \mathbb{N}$ is a constant to be determined later.

When changing G via a change ΔE with deletions E^- and insertions E^+, the dynamic program $\mathcal{P}$ handles deletions and insertions subsequently:

(1) Handling of deletions:
- **(a)** Define isolating weights w^- for $G^- \stackrel{\text{def}}{=} (V, E \setminus E^-)$. The weights w^- will differ from w only for the at most $\log^{c+d} n$ edges in E^-.
- **(b)** Compute an n^b-approximation of $(I - A_{(G^-,w^-)}(x))^{-1} \bmod 2$ using the existing n^b-approximation of $(I - A_{(G,w)}(x))^{-1} \bmod 2$.

(2) Handling of insertions:
- **(a)** Define a family $W^{-/+}$ of weightings such that one member of the family is isolating for $G^{-/+} \stackrel{\text{def}}{=} (V, (E \setminus E^-) \cup E^+)$. All weightings of the family will differ from w^- only for the at most $\log^{c+d} n$ edges in E^+.
- **(b)** Compute an n^b-approximation of $(I - A_{(G^{-/+},w^{-/+})}(x))^{-1} \bmod 2$ using the existing n^b-approximation of $(I - A_{(G^-,w^-)}(x))^{-1} \bmod 2$ for all members $w^{-/+}$ of $W^{-/+}$.

We first explain Steps (1a) and (2a) in more detail. For computing the isolating weights w^-, the program proceeds as follows. Skew-symmetric non-zero circulation weights u^- for G^- are obtained from the non-zero circulation weights u for G by setting the weight of deleted edges $e \in E^-$ to 0. As u^- gives the same weight to all simple cycles in G^- as u gives to these cycles in G, it has non-zero circulation. Now, the weight assignment w^- defined by $n^{k+2} + u^-(e)$ is isolating for G^- due to Lemma 8, and differs from w only for edges in E^-.

Computing the isolation weights for insertions is more challenging. In Lemma 11 below we show that from G^-, its transitive closure, and a set E^+ of edges of polylogarithmic size one can $\mathsf{FO}(\leq, +, \times)$-define a family $W^{-/+}$ of weight assignments such that one of these assignments is isolating for $G^{-/+}$.

For both Steps (1b) and (2b), the inverse of a matrix of polynomials over $\mathbb{Z}_2$ of polynomial degree needs to be updated after changing polylogarithmically many entries (i.e. entries corresponding to E^- and E^+, respectively). Inverses can be updated under such changes in

$\mathsf{FO}{+}\mathsf{Mod}\,2(\leq,+,\times)$ due to Lemma 10 (see below) and the observation that changes ΔA of size $\log^{c+d} n$ to such a matrix can be decomposed into UBV as required by Lemma 10, see Lemma 7 in [7]. For Step (2b) this is done in parallel for all members of $W^{-/+}$.

For checking whether there is a path from s to t after the change ΔE to G, the dynamic program checks whether there is a member of $W^{-/+}$ such that the s-t-entry of $(I - A_{(G^{-/+},w^{-/+})}(x))^{-1}$ mod 2 is non-zero. Since one member of $W^{-/+}$ is isolating, a path will be discovered this way. ◀

In the remainder of this subsection we show how inverses for matrices of polynomials can be updated under changes of polylogarithmic size, and how weights for inserted edges can be found.

The following lemma is obtained using the same techniques as in [7]. Here, $\mathbb{Z}_2[[x]]$ denotes the ring of formal power series with coefficients from $\mathbb{Z}_2$, and $\mathbb{Z}_2[x]$ denotes its subring that consists of all finite polynomials.

▶ **Lemma 10.** *Suppose $A \in \mathbb{Z}_2[[x]]^{n\times n}$ is invertible over $\mathbb{Z}_2[[x]]$, and $C \in \mathbb{Z}_2[x]^{n\times n}$ is an m-approximation of A^{-1}. If $A + \Delta A$ is invertible over $\mathbb{Z}_2[[x]]$ and ΔA can be written as UBV with $U \in \mathbb{Z}_2[x]^{n\times k}, B \in \mathbb{Z}_2[x]^{k\times k}$, and $V \in \mathbb{Z}_2[x]^{k\times n}$, then*

$$(A + \Delta A)^{-1} \approx_m C - CU(I + BVCU)^{-1}BVC$$

Furthermore, if $k \leq \log^c n$ for some fixed c and all involved polynomials have polynomial degree in n, then the right-hand side can be defined in $\mathsf{FO}{+}\mathsf{Mod}\,2(\leq,+,\times)$ from C and ΔA.

Proof sketch. The correctness of the equation can be proved exactly as in Proposition 14 in [7] (there, this is proved for $\mathbb{Z}[[x]]$ instead of $\mathbb{Z}_2[[x]]$).

We argue that the right-hand side can be defined in $\mathsf{FO}{+}\mathsf{Mod}\,2(\leq,+,\times)$. The involved matrix additions and multiplications modulo 2 can easily be expressed in $\mathsf{FO}{+}\mathsf{Mod}\,2(\leq,+,\times)$, see [11]. It remains to explain how the inverse of the $\log^c n \times \log^c n$ matrix $I + BVCU$ can be found.

To this end, recall that the i-j-entry of the inverse of a matrix D is equal to $(-1)^{i+j}\frac{\det D_{ji}}{\det D}$, where D_{ji} is obtained from D by removing the j-th row and the i-th column.

So, it is sufficient to show that the determinant of a $\log^c n \times \log^c n$ matrix of polynomials with polynomial degree can be expressed modulo 2. In [7, Lemma 15] it was shown that such a determinant can be expressed in $\mathsf{FO}{+}\mathsf{Maj}(\leq,+,\times)$, by observing that one only needs to be able to express the sum of polynomially many polynomials and the product of $\log^c n$ many polynomials. This observation is still valid for computing the determinant modulo 2 in $\mathsf{FO}{+}\mathsf{Mod}\,2(\leq,+,\times)$. Both kind of computations are possible modulo 2 in $\mathsf{FO}{+}\mathsf{Mod}\,2(\leq,+,\times)$ as well [11]. ◀

▶ **Lemma 11.** *Let $G = (V, E)$ be a graph and let $n = |V|$. Further, let w be a polynomially bounded isolating weight assignment for G, and let E^+ be a set of $\mathcal{O}(\log^c n)$ edges that is disjoint from E, for some $c \in \mathbb{N}$. Then there is a family W' of polynomially many polynomially bounded weight assignments such that*

1. *W' is $\mathsf{FO}(\leq,+,\times)$-definable from G, $\textsc{Reach}(G)$, E^+ and w,*
2. *all $w' \in W'$ agree with w on E,*
3. *at least one $w' \in W'$ is isolating for $(V, E \cup E^+)$.*

The proof works along the following lines. We use the approach from [18] to obtain weights for the inserted edges with the following idea: if there is an s-t-path that uses at least one inserted edge from E^+, then there is a unique minimal path under all s-t-paths

that use at least one such edge, where we ignore the weight of the paths that is contributed by edges from E. We multiply these constructed weights for the edges from E^+ by a large polynomial to ensure that the combined weight assignment with the existing weights for edges in E is isolating for the graph $(V, E \cup E^+)$.

The approach from [18] does not lead to polynomially bounded weights in the size of the graph it is used for. We construct them for a graph with $N = \mathcal{O}(\log^c n)$ many nodes, and although they are not polynomially bounded in N, they are in n.

We now get to the details of the construction. In the following, we consider graphs with two sets of edges. An *adorned* graph $G = (V, E, F)$ has, besides the set E of *real* edges, a further set F of *fictitious* edges, which is not necessarily disjoint from E. For each pair s, t of nodes, let $\mathcal{P}'_{s,t}$ be the set of s-t-paths in G that use at least one real edge $e \in E$ and arbitrarily many fictitious edges $e' \in F$. Let $\mathcal{P}_{s,t}$ be the set of edge sequences that result from $\mathcal{P}'_{s,t}$ by removing the fictitious edges from the paths.

We say that a weight assignment w *real-isolates* G, if (1) it maps each real edge $e \in E$ to a positive integer, and (2) each non-empty $\mathcal{P}_{s,t}$ has a unique minimal element under w. In the following, we will need a stronger property. We say that w *strongly* real-isolates G, if in addition for each pair $\mathcal{P}_{s,t}$ and $\mathcal{P}_{s',t'}$ of non-empty sets with $(s,t) \neq (s',t')$ the unique minimal elements of $\mathcal{P}_{s,t}$ and $\mathcal{P}_{s',t'}$ have different weights under w.

The following lemma can be proved along the lines of [18], see the full version for details.

▶ **Lemma 12.** *There is a constant $\beta \in \mathbb{N}$ such that for every natural number N and every adorned graph $G = (V, E, F)$ with $V = \{1, \dots, N\}$ there is a sequence $\bar{p} = p_1, p_2, \dots, p_{\log N}$ of primes, each consisting of at most $(\beta - 2) \log N$ bits, such that the weight assignment*

$$w_{\bar{p}}(e) = \begin{cases} \sum_{j=1}^{\log N} N^{\beta(\log N - j)}(w_0(e) \bmod p_j) & e \in E \\ 0 & e \in F \end{cases}$$

strongly real-isolates G. Here, $w_0(u,v) \stackrel{\text{def}}{=} 2^{(N+1)u+v}$.

Using this lemma, we can prove Lemma 11.

Proof of Lemma 11. Let $V_{\text{aff}} \subseteq V$ be the set of nodes with edges in E^+. We construct an adorned graph $H = (V_H, E_H, F_H)$ with node set $V_H \stackrel{\text{def}}{=} V_{\text{aff}}$ as follows. The set E_H of real edges is $E_H \stackrel{\text{def}}{=} E^+$, and the set F_H of fictitious edges is $F_H \stackrel{\text{def}}{=} \{(u,v) \mid u, v \in V_H, (u,v) \in \textsc{Reach}(G)\}$. So, a fictitious edge (u,v) of H represents the existence of a u-v-path in G.

Let $\beta, \bar{p}$ and $w_{\bar{p}}$ be as promised to exist for H by Lemma 12. Further, let n^k be the upper bound on the weights of w. We define the weight assignment w' for $G' \stackrel{\text{def}}{=} (V, E \cup E^+)$ as follows.

- $w'(e) = w(e)$ for all $e \in E$,
- $w'(e) = n^{k+2} \cdot w_{\bar{p}}(e)$ for all $e \in E^+ = E_H$.

We show that w' isolates G' first, afterwards we show that w' is a member of an $\mathsf{FO}(\leq, +, \times)$-definable family of weightings.

For showing that w' isolates G' suppose, towards a contradiction, that there are two lightest simple s-t-paths π, ρ in G' with respect to w', for some nodes s and t. Let $\pi_1\pi_2\pi_3 = \pi$ and $\rho_1\rho_2\rho_3 = \rho$ be the subpaths of π and ρ such that edges from E^+ are only used in π_2 and ρ_2 and those subpaths are minimal with that property. Notice that both π_2 and ρ_2 are non-empty, as otherwise ρ and π are also lightest paths in G with respect to w, contradicting the assumption that w isolates G. Let π' and ρ' be the paths in H that correspond to π_2 and ρ_2, where subpaths of π_2 and ρ_2 are replaced by fictitious edges. We consider two cases.

If $w_{\bar{p}}(\pi') \neq w_{\bar{p}}(\rho')$, then the total weight of π and ρ contributed by the edges from E^+ differs by at least n^{k+2}. As the total weight contributed by the remaining edges is upper-bounded by n^{k+1}, we have that $w'(\pi) \neq w'(\rho)$, the desired contradiction.

Thus assume, without loss of generality, that $w_{\bar{p}}(\pi') = w_{\bar{p}}(\rho')$. We can assume that both paths π' and ρ' are lightest paths in H: if, say, π' is not a lightest path, then we can replace π_2 in π by a path that uses lighter edges from E^+, leading to an overall lighter path by the argument of the previous case. Then, because $w_{\bar{p}}$ strongly isolates H, the edges from $E_H = E^+$ used in π' and ρ' must be equal, and the same is true for π_2 and ρ_2. These edges must also be used in the same order, as otherwise a path with fewer edges from E^+ exists, which by the argument of previous case is lighter than both π and ρ. Because π and ρ are different paths, there must be subpaths π^* and ρ^* that consist only of edges from E and are both simple u-v-paths, for some nodes u and v. As w is isolating for G, not both subpaths can be lightest u-v-paths in G. Say, ρ^* is not such a lightest path. If we replace ρ^* in ρ by the lightest u-v-path in G, we obtain a path that is lighter than ρ, as w' agrees with w on E. So, ρ is not a lightest s-t-path in G' with respect to w', the desired contradiction. It follows that w' is isolating for G'.

The weight assignment $w_{\bar{p}}$ is clearly $\mathsf{FO}(\leq, +, \times)$-definable from $G, \textsc{Reach}(G), E^+, w$ and $\bar{p}$, as H is $\mathsf{FO}(\leq, +, \times)$-definable, the involved numbers consist of at most polylogarithmically many bits, and $\mathsf{FO}(\leq, +, \times)$ can express the necessary arithmetic on numbers of that magnitude (see [14, Theorem 5.1]). The sequence $\bar{p}$ consists of $\mathcal{O}(\log \log n)$ many primes (as H is of size polylog) which in turn are represented by $\mathcal{O}(\log \log n)$ many bits, because H has only polylogarithmic size in n. So, $\bar{p}$ can be represented by a tuple of nodes from V, and it follows that a family W' of weight assignments with $w' \in W'$ is $\mathsf{FO}(\leq, +, \times)$-definable from $G, \textsc{Reach}(G), E^+$ and w. ◀

6.3 Computing weights for bounded-treewidth graphs

In this section, we show that isolating weights for graphs of bounded treewidth can be computed in LOGSPACE. As an immediate consequence, reachability can be maintained for such graphs under changes of polylogarithmic size.

A *tree decomposition* $\mathcal{T} = (T, \mathcal{B})$ of a graph $G = (V, E)$ consists of a (rooted, directed) tree $T = (I, F, r)$, with (tree) nodes I, (tree) edges F, a distinguished root node $r \in I$, and a function $\mathcal{B}\colon I \to 2^V$ such that

(1) the set $\{i \in I \mid v \in \mathcal{B}(i)\}$ is non-empty for each node $v \in V$,
(2) there is an $i \in I$ with $\{u, v\} \subseteq \mathcal{B}(i)$ for each edge $(u, v) \in E$, and
(3) the subgraph $T[\{i \in I \mid v \in \mathcal{B}(i)\}]$ is connected for each node $v \in V$.

We refer to the number of children of a node i of T as its *degree*, and to the set $\mathcal{B}(i)$ as its *bag*. We denote the parent node of i by $\text{parent}(i)$. The *width* of a tree decomposition is defined as the maximal size of a bag minus 1. The *treewidth* of a graph G is the minimal width among all tree decompositions of G. A tree decomposition is binary, if all tree nodes have degree at most 2. Its depth is the length of a longest path from the root r to a leaf of T. We inductively define the *height* $h(i)$ of i to be 1 if i is a leaf, and $h(i') + 1$ if i is an inner tree node and i' is a child of i with maximal height. For a node $v \in V$ we denote by $B(v)$ the highest bag that contains u, and let $h(u) \stackrel{\text{def}}{=} h(B(u))$. This bag $B(v)$ is well-defined for each node v thanks to condition (3) of the definition of a tree decomposition.

We usually identify tree nodes i and their bag $\mathcal{B}(i)$, and use the above notions and measures directly for bags. We also abuse notation and write $B \in \mathcal{B}$ if $B = \mathcal{B}(i)$ for some tree node i.

As a first step we construct isolation weights for bounded treewidth graphs that additionally have bounded degree. Isolation weights for all graphs of bounded treewidth are provided afterwards.

▶ **Proposition 13.** *Let $c, d, k \in \mathbb{N}$ be fixed. Let $G = (V, E)$ be a graph with maximal degree d, and let n be the number of its nodes. Let $\mathcal{T}$ be a binary tree decomposition of G with width k and depth at most $c \log n$. A polynomially bounded skew-symmetric weight assignment with non-zero circulation for G can be computed in* LOGSPACE.

Proof. The idea for assigning weights is the following. We associate each edge with one bag of the tree decomposition, namely the highest bag that contains one endpoint of the edge. For each bag B, we denote the set of all edges that are associated with B by $S(B)$. As the width of $\mathcal{T}$ and the degree of G are bounded by a constant, so is $|S(B)|$. An edge e is assigned a weight that depends exponentially on the height of the bag B it is associated with, and also exponentially on its position in some linear order on $S(B)$. For each cycle C there is a unique highest bag B_C that some of the cycle's edges is associated with. The idea for establishing non-zero circulation of C is that its weight is dominated by the weight of the unique edge which (1) is associated with B_C and (2) has largest index in the linear order on $S(B)$ among all edges of the cycle. As the height of a bag is logarithmic in n and $|S(B)|$ is bounded by a constant, the weight of every edge is polynomial in n.

We now proceed to the details. For each $e \in \vec{E}$, let B_e be the (unique) highest bag that contains one of the end points of e. For a bag B, define the set $S(B)$ of its associated edges as $S(B) \stackrel{\text{def}}{=} \{e \in \vec{E} \mid B = B_e\}$. Observe that the sets $S(B)$ partition the set $\vec{E}$ of edges and that the size of $S(B)$ is bounded by a constant $\beta \stackrel{\text{def}}{=} 2d(k+1)$, as each bag B contains at most $k+1$ nodes and each node has degree at most d in G and therefore degree at most $2d$ in $\vec{G}$. For each $S(B)$ we fix an enumeration of its elements[3]. Now, for each edge e, we set $h(e) = h(B_e)$ and $\ell(e) = i$, if e is the i-th element in the enumeration of $S(B_e)$.

We set the weight $w(e)$ of an edge $e = (u, v)$ with $u \leq v$ to be $w(e) \stackrel{\text{def}}{=} (4\beta \cdot 3^\beta + 2)^{h(e)} \cdot 3^{\ell(e)}$. The weight of an edge (u, v) with $u > v$ is $w(u, v) = -w(v, u)$. Notice that this weight assignment is polynomially bounded and skew-symmetric and can be computed in LOGSPACE. We now show that it has non-zero circulation.

Let $\mathcal{C}$ be any simple cycle in $\vec{G}$, and let $e_1, \ldots, e_m$ be an enumeration of its edges. Without loss of generality we assume that e_1 is the edge with the maximal weight among all edges in $\mathcal{C}$. This edge is well-defined, as there is a unique highest bag B such that $S(B)$ contains an edge of $\mathcal{C}$, and the term $4\beta \cdot 3^\beta + 2$ is strictly greater than $3^{\ell(e)}$ for any value of $\ell(e)$.

We show $|w(e_1)| > |w(e_2) + \cdots + w(e_m)|$, which implies the claim. Actually, we show that the weight of $w(e_1)$ exceeds the combined weight of all other edges e that are either in $S(B)$ and have $\ell(e) < \ell(e_1)$ or are in $S(B')$ for some bag B' below B in the tree decomposition. Note that there are $\sum_{h=1}^{h(e_1)-1} 2^{h(e_1)-h}$ many of those bags B', each $S(B')$ contains at most β edges, and the weight of each edge is upper bounded by $(4\beta \cdot 3^\beta + 2)^{h(B')} \cdot 3^\beta$.

$$
\begin{aligned}
&|w(e_2) + \cdots + w(e_m)| \\
&\quad < \sum_{i=1}^{l(e_1)-1} (4\beta \cdot 3^\beta + 2)^{h(e_1)} \cdot 3^i + \sum_{h=1}^{h(e_1)-1} 2^{h(e_1)-h} \cdot \beta \cdot (4\beta \cdot 3^\beta + 2)^h \cdot 3^\beta \\
&\quad = \sum_{i=1}^{l(e_1)-1} (4\beta \cdot 3^\beta + 2)^{h(e_1)} \cdot 3^i + \sum_{h=1}^{h(e_1)-1} 2^{h(e_1)} \cdot \beta \cdot (2\beta \cdot 3^\beta + 1)^h \cdot 3^\beta
\end{aligned}
$$

[3] As we devise an LOGSPACE algorithm, we can assume the existence of a linear order on the input.

$$= (4\beta \cdot 3^\beta + 2)^{h(e_1)} \cdot \sum_{i=1}^{l(e_1)-1} 3^i + 2^{h(e_1)} \cdot \beta \cdot 3^\beta \cdot \sum_{h=1}^{h(e_1)-1} (2\beta \cdot 3^\beta + 1)^h$$

$$= (4\beta \cdot 3^\beta + 2)^{h(e_1)} \cdot \frac{3^{l(e_1)} - 3}{2} + 2^{h(e_1)} \cdot \beta \cdot 3^\beta \cdot \frac{(2\beta \cdot 3^\beta + 1)^{h(e_1)} - (2\beta \cdot 3^\beta + 1)}{2\beta \cdot 3^\beta}$$

$$< (4\beta \cdot 3^\beta + 2)^{h(e_1)} \cdot 3^{\ell(e_1)} = |w(e_1)|$$ ◀

Non-zero circulation weights cannot only be computed in LOGSPACE for bounded-treewidth graphs with bounded degree, as given by Proposition 13, but also for all graphs with bounded treewidth. Also, using a result of Elberfeld, Jakoby and Tantau [10], no tree decomposition needs to be given as input.

▶ **Theorem 14.** *Let $k \in \mathbb{N}$ be fixed and let $G = (V, E)$ be a graph with treewidth at most k. A polynomially bounded skew-symmetric weight assignment with non-zero circulation for G can be computed in* LOGSPACE.

The idea for proving Theorem 14 is as follows. From a given graph G with treewidth at most k we construct a graph G' with treewidth and degree $\mathcal{O}(k)$ as well as a tree decomposition. The graph G' basically results from a tree decomposition $\mathcal{T}$ of G by making a copy of a node v for every bag of $\mathcal{T}$ that contains v. These copies are connected by an edge if the corresponding bags in $\mathcal{T}$ are. Using Proposition 13, we obtain non-zero circulation weights for G', and we show that they can be translated to non-zero circulation weights for G.

Proof. Fix $k \in \mathbb{N}$ and let $G = (V, E)$ be a graph with treewidth at most k. Let n be the size of V. There are constants $c_1, c_2 \in \mathbb{N}$ that only depend on k such that a binary tree decomposition $\mathcal{T} = (T, \mathcal{B})$ of G of width at most $c_1 k$ and depth at most $c_2 \log n$ can be computed in LOGSPACE [10]. From G and $\mathcal{T}$ we construct a graph $G' = (V', E')$ as follows. Let V' be the set $V' \stackrel{\text{def}}{=} \{v_B \mid B \in \mathcal{B}, v \in B\}$ and let $E' \stackrel{\text{def}}{=} \{(v_B, v_{B'}) \mid B' = \text{parent}(B)\} \cup \{(u_B, v_B) \mid (u, v) \in E, u \notin \text{parent}(B) \text{ or } v \notin \text{parent}(B)\}$. So, we have one copy v_B of a node $v \in V$ for each bag B such that v is contained in B. Two copies of a node are connected by an edge if they originate from adjacent bags in the tree decomposition, and there is an edge between two copies u_B and v_B, originating from the same bag B, if B is the highest bag of $\mathcal{T}$ that contains both endpoints u and v.

The degree of G' is bounded by $c_1 k + 3$. The tree decomposition $\mathcal{T}' = (T, \mathcal{B}')$ that replaces each bag B of $\mathcal{T}$ by $\{v_B \mid v \in B\} \cup \{v_{\text{parent}(B)} \mid v \in \text{parent}(B)\}$ is a tree decomposition of G' and has width at most $2c_1 k + 1$. Furthermore, it is binary and has depth at most $c_2 \log n$. So, by Proposition 13, one can compute in LOGSPACE polynomially bounded, skew-symmetric non-zero circulation weights w' for G'.

We construct a weight function w for $\vec{G}$ as follows. For that, we associate with each edge $(u, v) \in \vec{E}$ a sequence $P(u, v)$ of edges in $\vec{G'}$. Recall that for each edge (u, v) there is a highest bag in which both u and v appear. The bag above that bag contains either (a) none of the two vertices, or (b) v but not u, or (c) u but not v. The definition of $P(u, v)$ distinguishes these three cases:

1. Suppose $B(u) = B(v)$. We set $P(u, v) = (u_{B(u)}, v_{B(v)})$.
2. Suppose $B(u)$ is a proper descendant of $B(v)$. Let $B = B(u)$ and $B' = B(v)$. We set $P(u, v) = (u_B, v_B), (v_B, v_{\text{parent}(B)}), \ldots, (v_{\text{parent}(\cdots(\text{parent}(B)))}, v_{B'})$.
3. Suppose $B(v)$ is a proper descendant of $B(u)$. Let $B = B(u)$ and $B' = B(v)$. We set $P(u, v) = (u_B, u_{\text{parent}(\cdots(\text{parent}(B')))}), \ldots, (u_{\text{parent}(B')}, u_{B'}), (u_{B'}, v_{B'})$.

Now, let $w(u, v)$ be the sum $\sum_{e \in P(u,v)} w'(e)$ of the weights of the edges e in $P(u, v)$. Because w' is a polynomially bounded skew-symmetric weight assignment, so is w.

It remains to show that w has non-zero circulation. Let $\mathcal{C}$ be an arbitrary simple cycle in $\vec{G}$. We need to show that w assigns a non-zero weight to $\mathcal{C}$. Let $e_1, \ldots, e_m$ be the sequence of edges that constitutes $\mathcal{C}$, and let $\mathcal{W}'$ be the sequence $P(e_1), \ldots, P(e_m)$ of edges. By definition, the weight of $\mathcal{C}$ under w is the same as the weight of $\mathcal{W}'$ under w'.

Note that $\mathcal{W}'$ constitutes a cycle in $\vec{G'}$ which is not necessarily simple: some nodes might be visited more than once. We show that we can construct from $\mathcal{W}'$ a simple cycle $\mathcal{C}'$ by removing parts of $\mathcal{W}'$ with total weight 0. As a result, $\mathcal{C}'$ has the same weight as $\mathcal{W}'$ under w'. Because w' has non-zero circulation, the weight of $\mathcal{C}'$ and $\mathcal{W}'$ is non-zero, and so is the weight of $\mathcal{C}$ under w.

Suppose that some node u_B is visited twice by $\mathcal{W}'$. Then $\mathcal{W}'$ has the subsequence $P(v,u)P(u,v')$ for some nodes v and v', because u is visited only once in $\mathcal{C}$ and no node appears twice in a single sequence $P(u,u')$. Moreover, it most be that $h(u)$ is greater than both $h(v)$ and $h(v')$, or smaller than both $h(v)$ and $h(v')$, and either $B(v)$ is a descendent of $B(v')$ or $B(v')$ is a descendant of $B(v)$. We consider the case that $h(u)$ is greater than both $h(v)$ and $h(v')$, and $B(v')$ is a descendant of $B(v)$. The other cases are analogous. Then $P(v,u)P(u,v')$ visits the nodes $v_{B(v)}, u_{B(v)}, u_{\text{parent}(B(v))}, \ldots, u_{B(u)}, \ldots u_{\text{parent}(B(v))}, u_{B(v)}, \ldots, u_{B(v')}, v'_{B(v')}$ in that order. The closed walk from $u_{B(v)}$ to $u_{B(u)}$ and back to $u_{B(v)}$ has, because of skew-symmetry, a total weight of 0 under w'. So, the corresponding edges can be removed from $\mathcal{W}'$ without changing the weight. Repeating this step results in a simple cycle $\mathcal{C}'$ with the same weight under w' as $\mathcal{C}$ under w. As w' has non-zero circulation, the weight of $\mathcal{C}'$ is non-zero, and so is the weight of $\mathcal{C}$. ◀

7 Conclusion

The complexity of maintaining (variants of) the reachability query is the dominant research question in dynamic complexity theory. With this paper we basically settle this question for reachability in undirected graphs, at least with respect to the size of a change: reachability in undirected graphs is in $\mathsf{DynFO}(\leq, +, \times)$ if and only if the changes have at most polylogarithmic size. For reachability in directed graphs, we can only show this for insertions of polylogarithmic size, and the main open problem is whether this can be extended to also allow for deletions of single edges, non-constantly many edges, or even polylogarithmically many edges.

We give preliminary results for classes of graphs for which non-zero circulation weights can be computed in AC: reachability for these graphs is in $\mathsf{DynFO}{+}\mathsf{Mod}\,2(\leq, +, \times)$ under insertions and deletions of polylogarithmic size. We show that one can compute such weight assignments for graphs with bounded treewidth. Other graph classes for which this is possible include the class of planar graphs [23], and in general all graphs with bounded genus, which one can show using results from [5].

A question for further research is whether reachability for classes of directed graphs can be maintained in $\mathsf{DynFO}(\leq, +, \times)$ under insertions and deletions of polylogarithmic size. Candidate classes are graphs with bounded treewidth, and directed acyclic graphs.

References

1 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.

2 Xi Chen, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Near-optimal small-depth lower bounds for small distance connectivity. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 612–625. ACM, 2016. doi:10.1145/2897518.2897534.

3 Samir Datta, William Hesse, and Raghav Kulkarni. Dynamic complexity of directed reachability and other problems. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 356–367. Springer, 2014. doi:10.1007/978-3-662-43948-7_30.

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, Raghav Kulkarni, Raghunath Tewari, and N. V. Vinodchandran. Space complexity of perfect matching in bounded genus bipartite graphs. *J. Comput. Syst. Sci.*, 78(3):765–779, 2012. doi:10.1016/j.jcss.2011.11.002.

6 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.

7 Samir Datta, Anish Mukherjee, Nils Vortmeier, and Thomas Zeume. Reachability and distances under multiple changes. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 120:1–120:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.120.

8 Samir Datta, Anish Mukherjee, Nils Vortmeier, and Thomas Zeume. Reachability and distances under multiple changes. *CoRR*, abs/1804.08555, 2018. arXiv:1804.08555.

9 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.

10 Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of Bodlaender and Courcelle. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 143–152. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.21.

11 Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In Bruno Durand and Wolfgang Thomas, editors, *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, volume 3884 of *Lecture Notes in Computer Science*, pages 672–683. Springer, 2006. doi:10.1007/11672142_55.

12 Harold V Henderson and Shayle R Searle. On deriving the inverse of a sum of matrices. *Siam Review*, 23(1):53–60, 1981.

13 William Hesse. The dynamic complexity of transitive closure is in DynTC^0. *Theor. Comput. Sci.*, 296(3):473–485, 2003. doi:10.1016/S0304-3975(02)00740-5.

14 William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002. doi:10.1016/S0022-0000(02)00025-9.

15 Neil Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17(5):935–938, 1988. doi:10.1137/0217058.

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

17 Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

18 Vivek Anand T. Kallampally and Raghunath Tewari. Trading determinism for time in space bounded computations. In *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*, pages 10:1–10:13, 2016. doi:10.4230/LIPIcs.MFCS.2016.10.

19 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.

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

21 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82, 1987. doi:10.1145/28395.28404.

22 Róbert Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Inf.*, 26(3):279–284, 1988. doi:10.1007/BF00299636.

23 Raghunath Tewari and N. V. Vinodchandran. Green's theorem and isolation in planar graphs. *Inf. Comput.*, 215:1–7, 2012. doi:10.1016/j.ic.2012.03.002.

24 Nils Vortmeier. *Dynamic expressibility under complex changes*. PhD thesis, TU Dortmund University, Germany, 2019. doi:10.17877/DE290R-20434.