Reachability and Distances under Multiple Changes

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

Anish Mukherjee
Chennai Mathematical Institute, Chennai, India
anish@cmi.ac.in

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

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

Abstract

Recently it was shown that the transitive closure of a directed graph can be updated using first-order formulas after insertions and deletions of single edges in the dynamic descriptive complexity framework by Dong, Su, and Topor, and Patnaik and Immerman. In other words, Reachability is in DynFO.

In this article we extend the framework to changes of multiple edges at a time, and study the Reachability and Distance queries under these changes. We show that the former problem can be maintained in DynFO$(+,\times)$ under changes affecting $\mathcal{O}(\frac{\log n}{\log\log n})$ nodes, for graphs with n nodes. If the update formulas may use a majority quantifier then both Reachability and Distance can be maintained under changes that affect $\mathcal{O}(\log^c n)$ nodes, for fixed $c \in \mathbb{N}$. Some preliminary results towards showing that distances are in DynFO are discussed.

2012 ACM Subject Classification Theory of computation → Models of computation, Theory of computation → Finite Model Theory

Keywords and phrases dynamic complexity, reachability, distances, complex changes

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.120

Related Version A full version is available at [6], `https://arxiv.org/abs/1804.08555`.

Funding The authors acknowledge the financial support by the DAAD-DST grant "Exploration of New Frontiers in Dynamic Complexity". The first and the second authors were partially funded by a grant from Infosys foundation. The second author was partially supported by a TCS PhD fellowship. The last two authors acknowledge the financial support by DFG grant SCHW 678/6-2 on "Dynamic Expressiveness of Logics".

Acknowledgements We thank Rajit Datta and Partha Mukhopadhyay for pointing us to the Agrawal-Vinay technique for computing determinants. We also thank Thomas Schwentick for many illuminating discussions on dynamic complexity.

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 120; pp. 120:1–120:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

In today's databases, data sets are often large and subject to frequent changes. In use cases where only a fixed set of queries has to be evaluated on such data, it is not efficient to re-evaluate queries after each change, and therefore dynamic approaches have been considered. The idea is that when a database $\mathcal{D}$ is modified by changing a set $\Delta\mathcal{D}$ of tuples then the result of a query is recomputed by using its result on $\mathcal{D}$, the set $\Delta\mathcal{D}$, and possibly other previously computed auxiliary data.

One such dynamic approach is the *dynamic descriptive complexity* approach, formulated independently by Dong, Su, and Topor [8], as well as Patnaik and Immerman [19]. In their framework the query result and the auxiliary data are represented by relations, and updates of the auxiliary relations are performed by evaluating first-order formulas. The class of queries that can be maintained in this fashion constitutes the class DynFO. The motivation to use first-order logic as the vehicle for updates is that its evaluation is highly parallelizable and, in addition, that it corresponds to the relational algebra which is the core of SQL. Hence, if a query result can be maintained using a first-order update program, this program can be translated into equivalent SQL queries.

While it is desirable to understand how to update query results under complex changes $\Delta\mathcal{D}$, the focus of dynamic descriptive complexity so far has been on single tuple changes. The reason is that for many queries our techniques did not even suffice to tackle this case.

In recent years, however, we have seen several new techniques for maintaining queries. The Reachability query – one of the main objects of study in dynamic descriptive complexity – has been shown to be in DynFO using a linear algebraic method and a simulation technique [4]. The latter has been advanced into a very powerful tool: for showing that a query can be maintained in DynFO, it essentially suffices to show that it can be maintained for $\log n$ many change steps after initializing the auxiliary data by an AC^1 pre-computation[1] [5], where n is the size of the database's (active) domain. This tool has been successfully applied to show that all queries expressible in monadic second order logic can be maintained in DynFO on structures of bounded treewidth.

Those new techniques motivate a new attack on more complex changes $\Delta\mathcal{D}$. But what are reasonable changes to look at? Updating a query after a change $\Delta\mathcal{D}$ that replaces the whole database by a new database is essentially equivalent to the static evaluation problem with built-in relations: the stored auxiliary data has to be helpful for every possible new database, and therefore plays the role of built-in relations. Thus changes should be restricted in some way. Three approaches come to mind immediately: to only allow changes of restricted size; to restrict changes structurally; or to define changes in a declarative way.

In this article we focus on the first approach. Before discussing our results we shortly outline the other two approaches.

There is a wide variety of structural restrictions. For example, the change set $\Delta\mathcal{D}$ could only change the database locally or in such a way that the changes affect auxiliary relations only locally, e.g., if edges are inserted into distinct connected components it should be easier to maintain reachability. Another option is to restrict $\Delta\mathcal{D}$ to be of a certain shape, examples studied in the literature are cartesian-closed changes [8] and deletions of anti-chains [7].

A declarative mechanism for changing a database is to provide a set of parameterised rules that state which tuples should be changed depending on a parameter provided by a user. For example, a rule $\rho(x, y; z)$ could state that all edges (x, y) shall be inserted into a

[1] Readers not familiar with the circuit class AC^1 may safely think of LOGSPACE pre-computations.

graph such that x and y are connected to the parameter z. First-order logic as a declarative mean to change databases has been studied in [20], where it was shown that undirected reachability can be maintained under insertions defined by first-order formulas, and single tuple deletions.

In this article we study changes of small size with a focus on the Reachability and Distance queries. As can be seen from the discussion above, the former query has been well-studied in diverse settings of dynamic descriptive complexity, and therefore results on its maintainability under small changes serve as an important reference point.

There is another reason to study Reachability under non-constant size changes. Recall that Reachability is complete for the static complexity class NL. The result that Reachability is in DynFO does not imply NL $\subseteq$ DynFO, as DynFO is only known to be closed under very weak reductions, called bounded first-order reductions, under which Reachability is not NL-complete [19]. In short, these reductions demand that whenever a bit of an instance is changed, then only constantly many bits change in the image of the instance under the reduction. When a query such as Reachability is maintainable under larger changes, then this restriction may be relaxed and might yield new maintainability results for other queries under single edge changes.

In this work we show that Reachability can be maintained under changes of non-constant size. Since our main interest is the study of changes of non-constant size, we assume throughout the article that all classes come with built-in arithmetic and denote, e.g., by DynFO$(+,\times)$ the class of queries that can be maintained with first-order updates in the presence of a built-in linear addition and multiplication relations. How our results can be adapted to classes without built-in arithmetic is discussed towards the end of Section 3.

▶ **Theorem 1.** *Reachability can be maintained in* DynFO$(+,\times)$ *under changes that affect* $\mathcal{O}(\frac{\log n}{\log\log n})$ *nodes of a graph, where n is the number of nodes of the graph.*

The distance query was shown to be in DynFO+Maj by Hesse [13], where the class DynFO+Maj allows to specify updates with first-order formulas that may include majority quantifiers (equivalently, updates can be specified by uniform TC^0 computations). We generalize Hesse's result to changes of size polylogarithmic in the size of the domain.

▶ **Theorem 2.** *Reachability and Distance can be maintained in* DynFO+Maj$(+,\times)$ *under changes that affect* $\mathcal{O}(\log^c n)$ *nodes of a graph, where $c \in \mathbb{N}$ is fixed and n is the number of nodes of the graph.*

One of the important open questions of dynamic descriptive complexity is whether distances can be maintained in DynFO, even under single edge changes. We contribute to the solution of this question by discussing how distances can be maintained in a subclass of DynFO+Maj$(+,\times)$ that is only slightly stronger than DynFO$(+,\times)$.

Organization

After recapitulating notations in Section 2, we adapt the dynamic complexity framework to bulk changes in Section 3. Our main results, maintainability of reachability and distances under multiple changes, are proved in Section 4 and Section 5. We conclude with a discussion in Section 6.

2 Preliminaries

In this section we review basic definitions and results from finite model theory and databases.

We consider finite relational structures over relational signatures $\tau = \{R_1, \ldots, R_\ell\}$, where each R_i is a relational symbol of arity $\mathrm{Ar}(R_i)$. A τ-structure $\mathcal{D}$ consists of a finite domain D and relations $R_i^{\mathcal{D}}$ over D of arity $\mathrm{Ar}(R_i)$, for each $i \in \{1, \ldots, \ell\}$. The *active domain* $\mathrm{adom}(\mathcal{D})$ of a structure $\mathcal{D}$ contains all elements used in some tuple of $\mathcal{D}$. Since the motivation to study dynamic complexity originates from database theory, we use terminology from this area. In particular we use the terms "relational structure" and "relational database" synonymously.

We study the queries Reachability and Distance. Reachability asks, given a directed graph G, for all pairs s, t of nodes such that there is a path from s to t in G. Distance asks for the length of the shortest path between any pair of reachable nodes.

We assume familiarity with first-order logic FO and refer to [16] for an introduction. The logic FO+Maj extends FO by allowing majority quantifiers. Such quantifiers can ask whether more than half of all elements satisfy a given formula. We write FO(+,×) and FO+Maj(+,×) to denote that formulas have access to built-in relations $\leq, +, \times$ which are interpreted as linear order, addition and multiplication on the domain of the underlying structure. We note that FO(+,×) and FO+Maj(+,×) are equal to the circuit classes (DLOGTIME-)uniform AC^0 and TC^0, respectively [2].

In FO(+, ×), each tuple $(a_1, \ldots, a_c)$ encodes a number from $[n^c - 1]_0 \stackrel{\text{def}}{=} \{0, \ldots n^c - 1\}$. We will henceforth identify tuples over the domain and numbers.

It is well-known that FO(+, ×) supports arithmetic on numbers with polylog bits. Furthermore, iterated addition and multiplication for polylog many numbers with polylog bits can be expressed in FO(+, ×). More precisely:

▶ **Lemma 3** (cf. [14, Theorem 5.1]). *Suppose φ is a* FO(+, ×) *formula that defines $r \in \mathcal{O}(\log^c n)$ polylog bit numbers $a_1, \ldots, a_r$, then there are formulas ψ_+ and $\psi_\times$ that define the sum and product of $a_1, \ldots, a_r$, respectively.*

Due to these facts, many calculations can be defined in FO(+, ×). In particular, primes can be identified, and $\frac{\log n}{\log \log n}$ numbers of $\log \log n$ bits each can be encoded and decoded in $\log n$ bit numbers.

Suppose $p_1, \ldots, p_m$ are primes whose product is N. Then each number $A < N$ can be uniquely represented as a tuple $\bar{a} = (a_1, \ldots, a_m)$ where $a_i = A \mod p_i$. The tuple $\bar{a}$ is called *Chinese remainder representation* (CRR) of A. The number A can recovered from $\bar{a}$ via $A = \sum_i a_i h_i C_i - rN$, where $C_i = \frac{N}{m_i}$, h_i is the inverse of C_i modulo m_i, and $r = \sum_{i=1}^{m} \lfloor \frac{x_i h_i}{m_i} \rfloor$ [14, p. 702]. Due to Lemma 3, in FO(+, ×) one can encode and decode $\mathcal{O}(\log n)$ bit numbers into their CRR defined by $\mathcal{O}(\log n)$ primes with $\mathcal{O}(\log \log n)$ bits.

In this article we use basic notions and results from linear algebra which are introduced when they are needed. Throughout the article, a matrix with $\mathcal{O}(n^d)$ rows and columns and entries in $[n^c]_0$ will be represented by a relation R that contains a tuple $(\bar{r}, \bar{c}, \bar{v})$ if and only if the value at row $\bar{r}$ and column $\bar{c}$ is $\bar{v}$.

3 Dynamic Framework for Multiple Changes

We briefly repeat the essentials of dynamic complexity, closely following [21], and discuss generalisations due to changes of non-constant size.

The goal of a dynamic program is to answer a given query on an *input database* subjected to changes that insert or delete tuples. The program may use an auxiliary data structure represented by an *auxiliary database* over the same domain. Initially, both input and auxiliary database are empty; and the domain is fixed during each run of the program.

Changes. In previous work, changes of single tuples have been represented as explicit parameters for the formulas used to update the auxiliary relations. Non-constant size changes cannot be represented in this fashion. An alternative is to represent changes implicitly by giving update formulas access to the old input database as well as to the changed input database [11]. Here, we opt for this approach.

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

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 database. An *update rule* for updating an ℓ-ary auxiliary relation T after a change is a first-order formula φ over schema $\tau \cup \tau_{\text{aux}}$ with ℓ free variables, where τ_{aux} is the schema of the auxiliary database. After a change $\Delta\mathcal{D}$, the new version of T is $T \stackrel{\text{def}}{=} \{\vec{a} \mid (\mathcal{D} + \Delta\mathcal{D}, \mathcal{A}) \models \varphi(\vec{a})\}$ where $\mathcal{D}$ is the old input database and $\mathcal{A}$ is the current auxiliary database. Note that a dynamic program can choose to have access to the old input database by storing it in its auxiliary relations.

For a state $\mathcal{S} = (\mathcal{D}, \mathcal{A})$ of the dynamic program $\mathcal{P}$ with input database $\mathcal{D}$ and auxiliary database $\mathcal{A}$ 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 $\mathcal{Q}$ under changes that affect k elements (under changes of size k, respectively) if it has a q-ary auxiliary relation Q that at each point stores the result of $\mathcal{Q}$ applied to the current input database. More precisely, for each non-empty sequence α of changes that affect k elements (changes of size k, respectively), the relation Q in $\mathcal{P}_\alpha(\mathcal{S}_\emptyset)$ and $\mathcal{Q}(\alpha(\mathcal{D}_\emptyset))$ coincide, where $\mathcal{D}_\emptyset$ is an empty input structure, $\mathcal{S}_\emptyset$ is the auxiliary database with empty auxiliary relations over the domain of $\mathcal{D}_\emptyset$, and $\alpha(\mathcal{D}_\emptyset)$ is the input database after applying α.

If a dynamic program maintains a query, we say that the query is in DynFO. Similarly to DynFO one can define the class of queries DynFO$(+, \times)$ that allows for three particular auxiliary relations that are initialised as a linear order and the corresponding addition and multiplication relations. Other classes are defined accordingly.

For many natural queries $\mathcal{Q}$, in order to show that $\mathcal{Q}$ can be maintained, it is enough to show that the query can be maintained for a bounded number of steps. Intuitively, this is possible for queries for which isolated elements do not influence the query result, if there are many such elements. Formally, a query $\mathcal{Q}$ is *almost domain-independent* if there is a $c \in \mathbb{N}$ such that $\mathcal{Q}(\mathcal{A}) \restriction (\text{adom}(\mathcal{A}) \cup B) = \mathcal{Q}(\mathcal{A} \restriction (\text{adom}(\mathcal{A}) \cup B))$ for all structures $\mathcal{A}$ and sets $B \subseteq A \setminus \text{adom}(\mathcal{A})$ with $|B| \geq c$.

A query $\mathcal{Q}$ 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 database $\mathcal{D}$ 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 $\mathcal{Q}(\alpha(\mathcal{D}))$ coincide, where $\mathcal{S} = (\mathcal{I}, \mathbb{A}(\mathcal{I}, \leq))$.

The following theorem is a slight adaption of Theorem 3 from [5] and can be proved analogously.

▶ **Theorem 4.** *Every* $(\text{AC}^i, \log^i n)$*-maintainable, almost domain-independent query is in* DynFO$(+, \times)$.

The Role of the Domain and Arithmetic. In order to focus on the study of changes of non-constant size, we choose a simplified approach and include arithmetic in our setting. We state our results for DynFO(+, ×) and according classes to make it clear that we assume the presence of a linear order, addition and multiplication relation on the whole domain at all times.[2]

We shortly discuss the consequences of not assuming built-in arithmetic on our results. For single tuple changes, the presence of built-in arithmetic essentially gives no advantage.

▶ **Proposition 5** ([4, Theorem 4], formulation from [5, Proposition 2]). *If a query* $\mathcal{Q} \in$ DynFO(+, ×) *under single-tuple changes is almost domain-independent, then also* $\mathcal{Q} \in$ DynFO.

This result relies on the fact that one can maintain a linear order and arithmetic on the activated domain in DynFO under single-tuple changes [9], that is, on all elements that were in the active domain at some point of time. Under larger changes this is a priori not possible, as then one has to express in FO a linear order and arithmetic on the elements that enter the active domain.

An alternate approach to assuming the presence of built-in arithmetic is to demand that changes *provide* additional information on the changed elements, for example, that they provide a linear order and arithmetic on the domain of the change. Using this approach, our results can be stated in terms of DynFO and DynFO+Maj with the sole modification that sizes of changes are given relative to the size of the activated domain instead of with respect to the size of the whole domain. In this fashion our results also translate to the setting of first-order incremental evaluation systems of Dong, Su, and Topor [8], where the domain can grow and shrink.

4 Reachability under Multiple Changes

In this section we prove that Reachability can be maintained under multiple changes.

▶ **Theorem 1.** *Reachability can be maintained in* DynFO(+, ×) *under changes that affect* $\mathcal{O}(\frac{\log n}{\log \log n})$ *nodes of a graph, where* n *is the number of nodes of the graph.*

The approach is to use the well-known fact that Reachability can be reduced to the computation of the inverse of a matrix, and to invoke the Sherman-Morrison-Woodbury identity (cf. [12]) to update the inverse. This identity essentially reduces the update of inverses after a change affecting k nodes to the computation of an inverse of a $k \times k$ matrix.

The challenge is to define the updates in FO(+, ×). The key ingredients here are to compute inverses with respect to many primes, and throw away primes for which the inverse does not exist. As, by Theorem 4, it suffices to maintain the inverse for $\log^c n$ many steps for some c to be fixed later (see proof of Theorem 6), some primes remain valid if one starts from sufficiently – but polynomially – many primes. We show that the inverse of $k \times k$ matrices over $\mathbb{Z}_p$ can be defined in FO(+, ×) for $k = \frac{\log n}{\log \log n}$.

Theorem 1 in particular generalizes the result that Reachability can be maintained under single edge changes [4]; our proof is an alternative to the proof presented in the latter work.

[2] Different assumptions have been made in the literature. In [18], Patnaik and Immerman assume only a linear order to be present, while full arithmetic is assumed in [19]. Etessami observed that arithmetic can be built up dynamically, and therefore subsequent work usually assumed initially empty auxiliary relations, see e.g. [4, 5]. In the setting of first-order incremental evaluation systems usually no arithmetic is assumed to be present [8].

In [4], maintenance of Reachability is reduced to the question whether a matrix A has full rank, and it was shown that the rank can be maintained by storing and updating an invertible matrix B and a matrix D from which the rank can be easily extracted, such that $B \cdot A = D$.

4.1 Reachability and Matrix Inverses

There is a path from s to t in a graph G of size n with adjacency matrix A_G if and only if the s-t-entry of the matrix $(nI - A_G)^{-1}$ is non-zero. This follows from the equation $(nI - A_G)^{-1} = \frac{1}{n}\sum_{i=0}^{\infty}(\frac{1}{n}A_G)^i$ and the fact that A_G^i counts the number of paths from s to t of length i. Notice that $A \stackrel{\text{def}}{=} nI - A_G$ is invertible as matrix over $\mathbb{Q}$ for every adjacency matrix A_G since it is strictly diagonally dominant [15, Theorem 6.1.10].

When applying a change ΔG to G that affects k nodes, the adjacency matrix of G is updated by adding a suitable change matrix ΔA with at most k non-zero rows and columns to A. Thus Theorem 1 follows from the following proposition[3].

▶ **Theorem 6.** *When $A \in \mathbb{Z}^{n\times n}$ takes values polynomial in n and is assumed to stay invertible over $\mathbb{Q}$, then non-zeroness of entries of $A^{-1} \in \mathbb{Q}^{n\times n}$ can be maintained in* DynFO$(+,\times)$ *under changes that affect $\mathcal{O}(\frac{\log n}{\log\log n})$ rows and columns.*

Each change affecting $\mathcal{O}(\frac{\log n}{\log\log n})$ rows and columns can be partitioned into constantly many changes that affect $k \stackrel{\text{def}}{=} \frac{\log n}{\log\log n}$ rows and columns. We therefore concentrate on such changes in the following.

The change matrix ΔA for a change affecting k rows and columns has at most k non-zero rows and columns and can therefore be decomposed into a product UBV of suitable matrices U, B, and V, where U, B, and V have dimensions $n \times k$, $k \times k$, and $k \times n$, respectively.

▶ **Lemma 7.** *Fix a ring R. Suppose $M \in R^{n\times n}$ with non-zero rows $r_{i_1}, \dots, r_{i_k}$ and columns $c_{j_1}, \dots, c_{j_k}$. Then $M = UBV$ with $U \in R^{n\times k}, B \in R^{k\times k}$, and $V \in R^{k\times n}$ where*

1. *B is obtained from M by removing all-zero rows and columns.*
2. $U = \begin{pmatrix} \bar{u}_1 \\ \vdots \\ \bar{u}_n \end{pmatrix}$ *where* $\bar{u}_i = \begin{cases} \bar{0}^T & \text{if } i \notin \{i_1, \dots, i_k\} \\ \bar{e}_m^T & \text{if } i = i_m \end{cases}$
3. $V = \begin{pmatrix} \bar{v}_1, \dots, v_n \end{pmatrix}$ *where* $\bar{v}_j = \begin{cases} \bar{0} & \text{if } j \notin \{j_1, \dots, j_k\} \\ \bar{e}_m & \text{if } j = j_m \end{cases}$

Here, $\bar{e}_m$ denotes the m-th unit vector.

By the Sherman-Morrison-Woodbury identity (cf. [12]), the updated inverse can therefore be written as

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

The inverse of a matrix in $\mathbb{Z}^{n\times n}$ with entries that are polynomial in n is a matrix in $\mathbb{Q}^{n\times n}$ with entries $\frac{a}{b}$ that may involve numbers exponential in n. In particular computations cannot be performed in FO$(+,\times)$ directly. For this reason all computations will be done modulo many primes, and non-zeroness of entries of A^{-1} is extracted from these values.

[3] Due to lack of space some details are hidden here. The described reduction maps the empty graph to the matrix whose diagonal entries are n. Values of the inverse for this matrix cannot be determined in FO, and thus one does not immediately get the desired result for Reachability. This issue can be circumvented by mapping to matrices with only some non-zero entries on the diagonal, and studying the inverse of the matrices induced by non-zero diagonal entries.

Let us first see how to update $(A+\Delta A)^{-1}$ modulo a prime p under the assumption that both $A \pmod p$ and $A+\Delta A \pmod p$ are invertible. Observe that $(I+BVA^{-1}U)^{-1}$ is a $k \times k$ matrix and therefore an essential prerequisite to compute $(A+\Delta A)^{-1} \pmod p$ is to be able to define the inverse of such small matrices. That this is possible follows from the following lemma and the fact that $[D^{-1}]_{ij} = (-1)^{i+j}\frac{\det D_{ji}}{\det D}$ for invertible $D \in \mathbb{Z}_p^{k\times k}$. Here $[C]_{ij}$ denotes the ij-th entry of a matrix C and C_{ji} denotes the submatrix obtained by removing the j-th row and the i-th column.

▶ **Theorem 8.** *Fix a domain of size n and a prime $p \in \mathcal{O}(n^c)$. The value of the determinant of a matrix $A \in \mathbb{Z}_p^{k\times k}$ for $k = \frac{\log n}{\log\log n}$ can be defined in* $\mathrm{FO}(+,\times)$.

The technical proof of this theorem is deferred until the next Subsection 4.2.

That $(A+\Delta A)^{-1} \pmod p$ can defined in $\mathrm{FO}(+,\times)$ using Equation $(\star)$ now is a consequence of a straightforward analysis of the involved matrix operations.

▶ **Proposition 9.** *Fix a domain of size n and a prime $p \in \mathcal{O}(n^c)$. Given the inverse of a matrix $A \in \mathbb{Z}_p^{n\times n}$ and a matrix $\Delta A \in \mathbb{Z}_p^{n\times n}$ with at most $k = \frac{\log n}{\log\log n}$ non-zero rows and columns, one can determine whether $A+\Delta A$ is invertible in* $\mathrm{FO}(+,\times)$ *and, if so, the inverse can be defined.*

Proof. A decomposition of the matrix ΔA into UBV with $U \in \mathbb{Z}_p^{n\times k}, B \in \mathbb{Z}_p^{k\times k}$, and $V \in \mathbb{Z}_p^{k\times n}$ can be defined in $\mathrm{FO}(+,\times)$ using the characterization from Lemma 7. A simple analysis of the right hand side of Equation $(\star)$ – taking the dimensions of U, V, and B into account – yields that $VA^{-1}U$ and therefore $(I+BVA^{-1}U)^{-1}B$ are $k \times k$ matrices. Furthermore, $U(I+BVA^{-1}U)^{-1}BV$ is an $n \times n$ matrix that has at most k non-zero rows and columns.

The only obstacle to invertibility is that the inverse of $D \stackrel{\text{def}}{=} I+BVA^{-1}U$ may not exist in $\mathbb{Z}_p$. This is the case if and only if $\det(D) \equiv 0 \pmod p$ which can be tested using Theorem 8. If D is invertible, then its inverse can be defined by invoking Theorem 8 twice and using $[D]_{ij} = (-1)^{i+j}\frac{\det D_{ji}}{\det D}$.

Finally, if one knows how to compute $(I+BVA^{-1}U)^{-1}$, each entry in $A^{-1}U(I+BVA^{-1}U)^{-1}BV$ can be defined by adding k products of two numbers, and similarly for $(A^{-1}U(I+BVA^{-1}U)^{-1}BV)A^{-1}$. This can be done in $\mathrm{FO}(+,\times)$ due to Lemma 3. ◀

It remains to show how to maintain non-zeroness of entries of $(A+\Delta A)^{-1} \in \mathbb{Q}^{n\times n}$. Essentially a dynamic program can maintain a Chinese remainder representation of $(A+\Delta A)^{-1}$ and extract whether an entry is non-zero from this representation. An obstacle is that whenever $(I+BVA^{-1}U)^{-1} \pmod p$ does not exist for a prime p during the update process, then this prime p becomes *invalid* for the rest of the computation. The idea to circumvent this is simple: with each change, only a small number of primes become invalid. However, since the determinant can be computed in NC^2 (cf. [3]), using Theorem 4 we only need to be able to maintain a correct result for $\log^2 n$ many steps. Thus starting from sufficiently many primes will guarantee that enough primes are still valid after $\log^2 n$ steps.

We make these numbers more precise in the following.

Proof (of Theorem 6). By Theorem 4 and since values of the inverse of a matrix are almost domain-independent, it suffices to exhibit a dynamic program[4] that maintains non-zeroness of entries of A^{-1} for $\log^2 n$ changes of size $\frac{\log n}{\log\log n}$. The dynamic program maintains A^{-1}

[4] Actually we only describe a program that works correctly for sufficiently large n. However, small n can be easily dealt with separately.

(mod p) for each of the first $2n^3$ many primes p, which, by the Prime Number Theorem, can be found among the first n^4 numbers. Denote by P the set of the first $2n^3$ primes. The NC^2 initialization procedure computes $A^{-1} \pmod p$ for each prime in P. The update procedure for a change ΔA is simple:

(1) For each prime $p \in P$:
  (a) If $(A + \Delta A)^{-1} \pmod p$ is not invertible then remove p from P.
  (b) If $(A + \Delta A)^{-1} \pmod p$ is invertible then update $(A + \Delta A)^{-1} \pmod p$.

(2) Declare $[(A + \Delta A)^{-1}]_{st} \neq 0$ if there is a prime $p \in P$ with $[(A + \Delta A)^{-1}]_{st} \not\equiv 0 \pmod p$.

The Steps 1a and 1b can be performed in $\mathrm{FO}(+, \times)$ due to Proposition 9.

It remains to argue that the result from Step 2 is correct. Observe that the values of entries of A are at most n at all times, and therefore $\det(A) \leq n! n^n \leq 2^{n^2}$ for large enough n. Thus, since $\det(A) \neq 0$ over $\mathbb{Z}$ by assumption, there are at most n^2 primes p such that $\det(A) \equiv 0 \pmod p$, for all A reached after a sequence of changes.

In particular, $(A + \Delta A)^{-1} \pmod p$ is not invertible – equivalently, $(I + BVA^{-1}U)^{-1} \pmod p$ does not exist – for at most n^2 primes p. Hence, each time Step 1 is executed, at most n^2 primes are declared invalid and removed from P. All in all this step is executed at most $\log^2 n$ times, and therefore not more than n^3 primes are removed from P. Thus for the remaining n^3 valid primes, the inverses $(A + \Delta A)^{-1} \pmod p$ are computed correctly.

Each entry of $(A + \Delta A)^{-1}$ is, again, bounded by 2^{n^2}, so if $[(A + \Delta A)^{-1}]_{st} \neq 0$ there are at most n^2 primes $p \in P$ with $[(A + \Delta A)^{-1}]_{st} \equiv 0 \pmod p$. So, the result declared in Step 2 is correct. ◀

4.2 Defining the Determinant of Small Matrices

In this subsection we prove Theorem 8. The symbolic determinant of a $k \in \mathcal{O}(\frac{\log}{\log \log n})$ sized matrix is a sum of $k! \in n^{\mathcal{O}(1)}$ monomials and therefore cannot be naïvely defined in $\mathrm{FO}(+, \times)$. Here we use the fact that $\mathrm{FO}(+, \times)$ can easily convert $\log n$ bit numbers into their Chinese remainder presentation and back, and show how the determinant can be computed modulo $\log \log n$ bit primes.

It is easy to verify whether the value of a determinant modulo a $\mathcal{O}(\log \log n)$ bit prime is zero in $\mathrm{FO}(+, \times)$ by guessing a linear combination witnessing that the rank is less than full. We aim for a characterization that allows to reduce the verification of determinant values to such zeroness tests. To this end we use the self-reducibility and multilinearity of determinants. Assume $[A]_{11} \neq 0$ and that the determinant of A_{11} is also non-zero. Then the determinant can be written as $[A]_{11} \cdot d + r$ for some d and r. By finding an a such that the determinant is zero when $[A]_{11}$ is replaced by a in A we gain $r = -ad$. Repeating this step recursively for d – which is the determinant of a smaller matrix – one obtains a procedure for determining the value of the determinant that can be parallelized.

The following lemma is a preparation for deriving the characterization. We denote by A_i the matrix obtained from a matrix A by removing all rows and columns larger than i.

▶ **Lemma 10.** *Suppose $B = (\bar{b}_1, \ldots, \bar{b}_k) \in \mathbb{F}^{k \times k}$ is a non-singular matrix over a field $\mathbb{F}$. Then there is a permutation $\pi : [k] \to [k]$ such that for $A \stackrel{def}{=} (b_{\pi(1)}, \ldots, b_{\pi(k)})$:*

$$[A]_{ii} \neq 0 \text{ and } \det(A_i) \neq 0 \text{ for all } i \in [k]$$

The following proposition characterizes the determinant of a matrix. We will see that this characterization allows for parallel computation of the determinant of small matrices.

▶ **Proposition 11.** *Suppose* $A = (a_{ij})_{1 \leq i,j \leq k} \in \mathbb{F}^{k \times k}$ *is a matrix over a field* $\mathbb{F}$ *such that* $a_{ii} \neq 0$ *and* $\det(A_i) \neq 0$ *for all* $i \in [k]$*. Let* A_i^b *be the matrix obtained from* A_i *by replacing* a_{ii} *by* b *for some* $b \in \mathbb{F}$*. Then there are unique* $b_2, \ldots, b_k \in \mathbb{F}$ *and* $d_1, \ldots, d_k \in \mathbb{F}$ *such that*

1. $d_1 = a_{11}$,
2. $d_i = (a_{ii} - b_i)d_{i-1}$, *and*
3. $\det(A_i^b) = 0$

for $2 \leq i \leq k$*. Furthermore, it holds that* $d_i = \det(A_i)$.

Finally we show that the characterization from the previous proposition can be used to define the determinant of small matrices in $\mathrm{FO}(+, \times)$.

Proof (of Theorem 8). Suppose $A \in \mathbb{Z}_p^{k \times k}$ is a matrix with $p \in \mathcal{O}(n^c)$ and $k = \frac{\log n}{\log \log n}$. The idea is to define $\det(A) \pmod p$ in Chinese remainder representation for primes $q_1, \ldots, q_m$. A simple calculation shows that $m \in \mathcal{O}(\log n)$ primes each of $\mathcal{O}(\log \log n)$ bits suffice. The Chinese remainder representation can be defined from A and the value $\det(A) \pmod p$ can be recovered from the values $\det(A) \pmod{q_1}, \ldots, \det(A) \pmod{q_m}$ in $\mathrm{FO}(+, \times)$ due to Lemma 3. Thus let us show how to define $\det(A) \pmod q$ for a prime q of $\mathcal{O}(\log \log n)$ bits.

The idea is to first test whether the determinant is zero. If not, the fact that it is not zero is used to define the determinant using Proposition 11.

If $A \pmod q$ is singular then there exists a non-trivial linear combination of the columns that yields the all zero vector. Such a linear combination is determined by specifying one $\mathcal{O}(\log \log n)$ bit number for each of the k columns. It can thus be encoded in $\mathcal{O}(\log n)$ bits, and therefore existentially quantified by a first-order formula. Such a "guess" can be decoded (i.e., the k numbers of $\mathcal{O}(\log \log n)$ length can be extracted) in $\mathrm{FO}(+, \times)$, see Section 2. Checking if a guessed linear combination is zero requires to sum k small numbers and is hence in $\mathrm{FO}(+, \times)$ due to Lemma 3.

Now, for defining the determinant $\det(A) \pmod q$ when $A \pmod q$ is non-singular, a formula can guess a permutation π of $[k]$ and verify that it satisfies the conditions from Lemma 10. Note that such a permutation can be represented as a sequence of k pairs of numbers of $\log \log n$ bits each, and hence be stored in $\mathcal{O}(\log n)$ bits. The verification of the conditions from Lemma 10 requires the zero-test for determinants explained above. After fixing π, the values $b_2, \ldots, b_k$ as well as $d_1, \ldots, d_k$ from Proposition 11 can be guessed and verified. Again, these numbers can be stored in $\mathcal{O}(\log n)$ bits. For verifying the conditions from Proposition 11 on the determinants of A_i^b, the zero-test for determinants is used. ◀

5 Distances under Multiple Changes

In this section we extend the techniques from the previous section to show how distances can be maintained under changes that affect polylogarithmically many nodes with first-order updates that may use majority quantifiers. Afterwards we discuss how the techniques extend to other dynamic complexity classes.

▶ **Theorem 2.** *Reachability and Distance can be maintained in* DynFO+Maj$(+, \times)$ *under changes that affect* $\mathcal{O}(\log^c n)$ *nodes of a graph, where* $c \in \mathbb{N}$ *is fixed and* n *is the number of nodes of the graph.*

The idea is to use generating functions for counting the number of paths of each length, following Hesse [13]. Fix a graph G with adjacency matrix $A_G \in \mathbb{Z}^{n \times n}$ and a formal variable x. Then $D \stackrel{\text{def}}{=} \sum_{i=0}^{\infty} (xA_G)^i$ is a matrix of formal power series from $\mathbb{Z}[[x]]$ such that if $[D]_{st} = \sum_{i=0}^{\infty} c_i x^i$ then c_i is the number of paths from s to t of length i. In particular, the distance between s and t is the smallest i such that c_i is non-zero. Note that if such an i exists, then $i < n$.

Similarly to the corresponding matrix from the previous section, the matrix D is invertible over $\mathbb{Z}[[x]]$ and can be written as $(I - xA_G)^{-1}$ (cf. [10, Example 3.6.1]). The maintenance of distances thus reduces to maintaining for a matrix $A \in \mathbb{Z}[[x]]$, for each entry (s,t), the smallest $i < n$ such that the ith coefficient is non-zero.

▶ **Theorem 12.** *Suppose $A \in \mathbb{Z}[[x]]^{n\times n}$ stays invertible over $\mathbb{Z}[[x]]$. For all $s,t \in [n]$ one can maintain the smallest $i < n$ such that the ith coefficient of the st-entry of A^{-1} is non-zero in* DynFO+Maj$(+,\times)$ *under changes that affect $\mathcal{O}(\log^c n)$ nodes, for fixed $c \in \mathbb{N}$.*

The idea is the same as for Reachability. When updating A to $A + \Delta A$ then one can decompose the change matrix ΔA into UBV for suitable matrices U, B, and V, and apply the Sherman-Morrison-Woodbury identity ($\star$), this time over the field of fractions $\mathbb{Z}((x))$.

Of course computing with inherently infinite formal power series is not possible in DynFO+Maj$(+,\times)$. However, as stated in Theorem 12, in the end we are only interested in the first $i < n$ coefficients of power series. We therefore show that it suffices to truncate all occurring power series at the n-th term and use FO+Maj$(+,\times)$'s ability to define iterated sums and products of polynomials [14].

Formally, we have to show that no precision for the first $i < n$ coefficients is lost when computing with truncated power series. This motivates the following definition. A formal power series $g(x) = \sum_i c_i x^i \in \mathbb{Z}[[x]]$ is an *m-approximation* of a formal power series $h(x) = \sum_i d_i x^i \in \mathbb{Z}[[x]]$, denoted by $g(x) \approx_m h(x)$, if $c_i = d_i$ for all $i \leq m$. This notion naturally extends to matrices over $\mathbb{Z}[[x]]$: a matrix $A \in \mathbb{Z}[[x]]^{\ell\times k}$ is an m-approximation of a matrix $B \in \mathbb{Z}[[x]]^{\ell\times k}$ if each entry of A is an m-approximation of the corresponding entry of B. The notion of m-approximation is preserved under all arithmetic operations that will be relevant.

▶ **Lemma 13.** *Fix an $m \in \mathbb{N}$.*

1. *Suppose $g(x), g'(x), h(x), h'(x) \in \mathbb{Z}[[x]]$ with $g(x) \approx_m g'(x)$ and $h(x) \approx_m h'(x)$. Then*
   - **(i)** *$g(x) + h(x) \approx_m g'(x) + h'(x)$,*
   - **(ii)** *$g(x)h(x) \approx_m g'(x)h'(x)$, and*
   - **(iii)** *$\frac{1}{g(x)} \approx_m \frac{1}{g'(x)}$ whenever $g(x)$ and $g'(x)$ are normalized.*
2. *Suppose $A, A', B, B' \in \mathbb{Z}[[x]]^{n\times n}$ with $A \approx_m A'$ and $B \approx_m B'$. Then*
   - **(i)** *$A + B \approx_m A' + B'$,*
   - **(ii)** *$AB \approx_m A'B'$,*
   - **(iii)** *If A is invertible over $\mathbb{Z}[[x]]$ then so is A', and $A^{-1} \approx_m A'^{-1}$.*

Here, a formal power series $\sum_i c_i x^i \in \mathbb{Z}[[x]]$ is *normalized* if $c_0 = 1$.

An approximation of the inverse of a matrix $A \in \mathbb{Z}[[x]]^{n\times n}$ can be updated using the Sherman-Morrison-Woodbury identity.

▶ **Proposition 14.** *Suppose $A \in \mathbb{Z}[[x]]^{n\times n}$ is invertible over $\mathbb{Z}[[x]]$, and $C \in \mathbb{Z}[[x]]^{n\times n}$ is an m-approximation of A^{-1}. If $A + \Delta A$ is invertible over $\mathbb{Z}[[x]]$ and ΔA can be written as UBV with $U \in \mathbb{Z}[[x]]^{n\times k}, B \in \mathbb{Z}[[x]]^{k\times k}$, and $V \in \mathbb{Z}[[x]]^{k\times n}$, then*

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

Proof. This follows immediately from the Sherman-Morrison-Woodbury identity $(A + UBV)^{-1} = A^{-1} - A^{-1}U(I + BVA^{-1}U)^{-1}BVA^{-1}$ and Lemma 13. ◀

As already discussed in Section 4, the Sherman-Morrison-Woodbury identity involves inverting $k \times k$ matrices, which reduces to computing the determinant of such matrices. We show that this is possible in FO+Maj for $k \times k$ matrices of polynomials for $k \in \mathcal{O}(\log^c n)$.

▶ **Lemma 15.** *Fix a domain of size n and $c \in \mathbb{N}$. The determinant of a matrix $A \in \mathbb{Z}[x]^{k\times k}$, with entries of degree polynomial in n, can be defined in* FO+Maj$(+,\times)$ *for $k \in \mathcal{O}(\log^c n)$.*

Proof. We show that the value can be computed in uniform TC^0, which is as powerful as FO+Maj$(+,\times)$ [2].

Computing the determinant of an $k \times k$ matrix is equivalent to computing the iterated matrix product of k matrices of dimension at most $(k+1)\times(k+1)$ [3], and this reduction is a uniform TC^0-reduction as can be seen implicitly in [17, p. 482]. Thus the lemma statement follows from the fact that iterated products of matrices $A_1, \ldots, A_k \in \mathbb{Z}[x]^{k\times k}$ with $k \in \mathcal{O}(\log^c n)$ can be computed in uniform TC^0, which can be proven like in [1, p. 69]. ◀

Proof (of Theorem 12). The dynamic program maintains an n-approximation $C \in \mathbb{Z}[x]^{n\times n}$ of A^{-1} that truncates A^{-1} at degree n. When A is updated to $A + \Delta A$ then:

1. ΔA is decomposed into suitable $U \in \mathbb{Z}[x]^{n\times k}$, $B \in \mathbb{Z}[x]^{k\times k}$, and $V \in \mathbb{Z}[x]^{k\times n}$;
2. C is updated via $C' \stackrel{\text{def}}{=} C - CU(I + BVCU)^{-1}BVC$;
3. All entries of C' are truncated at degree n.

The steps can be defined in FO+Maj$(+,\times)$ due to Lemma 7, Lemma 15, and the fact that iterated addition and multiplication of polynomials can be defined in FO+Maj$(+,\times)$, see [14]. The maintained matrix C is indeed an n-approximation of A^{-1} due to Proposition 14. ◀

From the proof of Theorem 12 it is clear that the main obstacle towards maintaining distances for changes that affect a larger set of nodes is to compute determinants of larger matrices. Since distances can be computed in NL, only classes below NL are interesting from a dynamic perspective. As an example we state a result for the circuit class NC^1.

▶ **Corollary 16.** *Reachability and Distance can be maintained in* DynNC^1 *under changes that affect $\mathcal{O}(2^{\sqrt{\log n/\log^* n}})$ nodes.*

Here $\log^* n$ denotes the smallest number i such that i-fold application of log yields a number smaller than 1.

6 Conclusion

For us it came as a surprise that Reachability can be maintained under changes of non-constant size, without any structural restrictions. In contrast, the dynamic program for Reachability from [4] can only deal with changing $\log n$ many outgoing edges of single nodes (or, symmetrically, $\log n$ many incoming edges; a combination is not possible).

It would be interesting to improve our results for Dyn FO$(+,\times)$ to changes of size $\mathcal{O}(\log n)$. The obstacle is the computation of determinants of matrices of this size, which we can only do for $\mathcal{O}(\frac{\log n}{\log\log n})$ size matrices. Yet in principle our approach can deal with certain changes that affect more nodes: the matrices U and V in the Sherman-Morrison-Woodbury identity can be chosen differently, as long as all computations involve only adding $\mathcal{O}(\log n)$ numbers.

One of the big remaining open questions in dynamic complexity is whether distances are in Dyn FO. Our approach sheds some light. It can be adapted so as to maintain information within Dyn FO$(+,\times)$ from which shortest distances can be extracted in FO+Maj$(+,\times)$.

▶ **Theorem 17.** *Distances can be defined by a* FO+Maj$(+,\times)$ *query from auxiliary relations that can be maintained in* Dyn FO$(+,\times)$ *under changes that affect $\mathcal{O}(\frac{\log n}{\log\log n})$ nodes.*

References

1 Manindra Agrawal and V Vinay. Arithmetic circuits: A chasm at depth four. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 67–75. IEEE, 2008.

2 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.

3 Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(1-3):2–21, 1985. doi:10.1016/S0019-9958(85)80041-3.

4 Samir Datta, Raghav Kulkarni, Anish Mukherjee, Thomas Schwentick, and Thomas Zeume. Reachability is in DynFO. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 2015. doi:10.1007/978-3-662-47666-6.

5 Samir Datta, Anish Mukherjee, Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. A strategy for dynamic programs: Start over and muddle through. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 98:1–98:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. URL: http://www.dagstuhl.de/dagpub/978-3-95977-041-5, doi:10.4230/LIPIcs.ICALP.2017.98.

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

7 Guozhu Dong and Chaoyi Pang. Maintaining transitive closure in first order after node-set and edge-set deletions. *Inf. Process. Lett.*, 62(4):193–199, 1997. doi:10.1016/S0020-0190(97)00066-5.

8 Guozhu Dong, Jianwen Su, and Rodney W. Topor. Nonrecursive incremental evaluation of datalog queries. *Ann. Math. Artif. Intell.*, 14(2-4):187–223, 1995. doi:10.1007/BF01530820.

9 Kousha Etessami. Dynamic tree isomorphism via first-order updates. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 235–243, 1998. doi:10.1145/275487.275514.

10 Chris D. Godsil. *Algebraic combinatorics*. Chapman and Hall mathematics series. Chapman and Hall, 1993.

11 Erich Grädel and Sebastian Siebertz. Dynamic definability. In Alin Deutsch, editor, *15th International Conference on Database Theory, ICDT '12, Berlin, Germany, March 26-29, 2012*, pages 236–248. ACM, 2012. URL: http://dl.acm.org/citation.cfm?id=2274576, doi:10.1145/2274576.2274601.

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 Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

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

17 Meena Mahajan and V. Vinay. Determinant: Old algorithms, new insights. *SIAM J. Discrete Math.*, 12(4):474–490, 1999. doi:10.1137/S0895480198338827.

18 Sushant Patnaik and Neil Immerman. Dyn-fo: A parallel, dynamic complexity class. In Victor Vianu, editor, *Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 24-26, 1994, Minneapolis, Minnesota, USA*, pages 210–221. ACM Press, 1994. URL: http://dl.acm.org/citation.cfm?id=182591, doi:10.1145/182591.182614.

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, Nils Vortmeier, and Thomas Zeume. Dynamic complexity under definable changes. In Michael Benedikt and Giorgio Orsi, editors, *20th International Conference on Database Theory, ICDT 2017, March 21-24, 2017, Venice, Italy*, volume 68 of *LIPIcs*, pages 19:1–19:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. URL: http://www.dagstuhl.de/dagpub/978-3-95977-024-8, doi:10.4230/LIPIcs.ICDT.2017.19.

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