An implementation of dynamic complexity

Nils Vortmeier

Abstract

A dynamic program, as introduced by Dong, Su and Topor and Patnaik and Immerman, maintains the result of a fixed query for an input database which is subject to changes. It can use an auxiliary database whose relations are updated via first-order formulas upon changes of the input database. Important queries like the reachability query can be maintained dynamically, despite the fact that they cannot be expressed in first-order logic in the static setting. However, there is no recent practical evaluation of the performance of dynamic programs based on these results.

In this talk I will present preliminary evaluation results on a prototypical implementation in SQL. We study the reachability query for undirected graphs. In addition to insertions and deletions of single edges, we also test the performance for insertions of many edges that are defined by (unions of) conjunctive queries.

This talk is based on joint work with Thomas Schwentick, Thomas Zeume and Dennis Ciba.

Modern data management systems handle an enormous amount of data, but still have to react quickly to requests. If a query has to be answered frequently, it is often more efficient to materialize the query result instead of recomputing the query from scratch every time the result is requested. However, when the underlying data changes, the stored query result possibly becomes invalid. In that case, instead of evaluating the query from scratch, we *update* the query result with the help of the old result and possibly further previously computed auxiliary data. This auxiliary data then also needs to be updated whenever the database changes.

This dynamic approach of maintaining information under changes was formalized from a descriptive point of view by Patnaik and Immerman [6] and, independently, by Dong, Su and Topor [4]. Within the framework of dynamic (descriptive) complexity, for a relational database subject to changes and a query $\mathcal{Q}$, a *dynamic program* maintains auxiliary relations including a distinguished relation representing the result of $\mathcal{Q}$. If a change to the database occurs, the auxiliary relations are updated by first-order formulas (or, equivalently, core SQL queries). We call the class of queries that can be maintained in this way DynFO.

Recently there has been considerable progress in maintaining important queries. If changes are insertions or deletions of single edges, the reachability query on arbitrary graphs can be maintained in DynFO [1], as well as every MSO-definable query on graphs with bounded treewidth [2].

Furthermore, progress has been obtained in maintaining queries under more complex changes. In the literature, most dynamic programs restrict the admissible changes to insertions and deletions of single tuples, although in practical database applications, changes of a database are often defined by queries that are applied to the current instance. For example, using an SQL statement, one can insert all edges (v, w) such that there are edges (w, u) and (u, v) already present, for some node u. In [7], we studied changes to the database that are defined by first-order formulas (or restrictions thereof) and reviewed which queries can still be maintained in this setting. It turned out that the reachability query on undirected graphs can still be maintained in DynFO if changes are first-order defined insertions of edges and deletions of single edges, which extends results of [6, 3].

This progress is a strong motivation to evaluate the approach in practical scenarios. In this talk, we present a prototypical implementation in SQL of dynamic programs for maintaining undirected reachability under changes of single edges following [3], and under defined insertions and single-edge deletions. For the latter, we focus on insertions that are defined by (unions of) conjunctive queries, possibly with negations, which allow for a more efficient implementation than full first-order logic in general. We give preliminary results of an evaluation that compares the performance of these programs with a dynamic program presented in [5] and with recomputation of the query result from scratch.

This talk is based on joint work with Thomas Schwentick, Thomas Zeume and Dennis Ciba.

References

[1] 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.

[2] Samir Datta, Anish Mukherjee, Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. A strategy for dynamic programs: Start over and muddle through. In *ICALP*, 2017. Accepted for publication.

[3] 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.

[4] Guozhu Dong, Jianwen Su, and Rodney Topor. Nonrecursive incremental evaluation of datalog queries. *Annals of Mathematics and Artificial Intelligence*, 14, 1995.

[5] Chaoyi Pang, Guozhu Dong, and Kotagiri Ramamohanarao. Incremental maintenance of shortest distance and transitive closure in first-order logic and SQL. *ACM Trans. Database Syst.*, 30(3):698–721, 2005.

[6] Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel, dynamic complexity class. *J. Comput. Syst. Sci.*, 55(2):199–209, 1997.

[7] 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.