Incremental Graph Transformation in Relational Databases

Gergely Varró[*]

The theory of graph transformation [2] was originally developed as a generalization of Chomsky grammars from strings to graphs. Methods, techniques, and tools from the area of graph transformations have already been studied and applied in many fields of computer science such as formal language theory, pattern recognition and generation, compiler construction, software engineering, etc.

Despite the large variety of existing graph transformation tools, the implementation of their graph transformation engine typically follows the same principle. In this respect, first a matching occurrence of the left-hand side (LHS) of the graph transformation rule is being found by some sophisticated graph pattern matching algorithm. Then the engine performs some local modifications to add or remove graph elements to the matching pattern, and the entire process starts all over again.

Since graph pattern matching leads to the subgraph isomorphism problem that is known to be NP-complete in general, this step is considered to be the most crucial in the overall performance of a graph transformation engine. Current tools (e.g., PROGRES [4]) use different efficient strategies for the graph pattern matching phase.

However, I argue that the overall complexity of a graph transformation engine is not necessarily equal to the complexity of the graph pattern matching phase, especially for long transformation sequences. During the execution of a transformation step, instance models are only modified locally in the context of the matching occurrence of the LHS and the rest of the instance model left unchanged. Thus any implementation that does not reuse the information collected for the previous matching and that restarts the complex and expensive pattern matching phase from scratch is not optimal.

I propose a technique based on *incremental updates* [3] which, in itself, is not a new idea, since it has been widely accepted and successfully used in relational databases, but it provides a new philosophy for implementing efficient graph transformation engines.

The main idea of incremental updates in graph transformation systems is to keep track of all possible matchings of graph transformation rules (e.g., in database tables) to make the graph pattern matching step very fast. Afterwards when a rule is applied we update data in all locations it is required. Since graph transformation typically manipulates only a small fragment of the instance model, incremental updates require minor changes to the stored data. Naturally, the initialization phase needs some considerable amount of pre-processing prior to the transformation, but the subsequent transformation process itself becomes much faster.

Relational databases give several supporting features (e.g., foreign key constraints, database views) for preserving data consistency. A *view* is a query that defines a computed relation in the database (see [1] for details). It is updated incrementally in most off-the-shelf relational databases. Our idea is to build a graph transformation engine on the top of a relational database to exploit various results of database theory.

References

1. A. Gupta and I. S. Mumick. Maintenance of materialized views: Problems, techniques and applications. *IEEE Quarterly Bulletin on Data Engineering; Special Issue on Materialized Views and Data Warehousing*, 1995.
2. G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 1: Foundations*. World Scientific, 1997.
3. G. Varró and D. Varró. Graph transformation with incremental updates. In *Proc. 4th Int. Workshop on Graph Transformation and Visual Modeling Techniques*, 2004.
4. A. Zündorf. Graph pattern-matching in PROGRES. In *Proc. 5th Int. Workshop on Graph Grammars and their Application to Computer Science*, volume 1073 of *LNCS*, pages 454–468. Springer-Verlag, 1996.

[*] Email: `gervarro@cs.bme.hu`