Graphical Representation and Graph Transformation

Hartmut Ehrig and Gabriele Taentzer
Technical University of Berlin
E-mail: ehrig@cs.tu-berlin.de, gabi@cs.tu-berlin.de

1. RELEVANCE IN COMPUTER SCIENCE

Graphical representation are extremely useful to illustrate complex structures in a direct and intuitive way, and as such they are widely used in many fields of Computer Science; examples of well-known graph-like structures include control- and data-flow tables. entity relationship diagrams, Petri nets, visualization of hardware and software architectures, evolution diagrams of non-deterministic processes, SADT-diagrams, state charts, and many more. The underlying structure of all kinds of graphical representation are graphs, where nodes and edges represent structural elements and relations between these elements. Moreover attributes of nodes and edges describe layout, contents or other annotations of these elements and relations.

Graph transformations are used to describe dynamic changes of graphical representations. Graph grammars and graph transformation systems allow a rule-based manipulation of graphs. Graphs rules describe actions in if-then manner: if the left-hand side matches with a subgraph of the current working graph, it can be replaced by inserting the right-hand side of the rule instead. In comparison with string grammars or term rewriting embedding of the right-hand side is not automatically clear, for embedding a certain part of the rewriting context has to be considered.

Graph grammars and graph transformation systems have been shown since their origin in the late sixties to have fruitful applications in various fields of computer science, including formal language theory, pattern recognition and generation, compiler construction, implementation of functional languages, software engineering supporting view-based system development, visual language definition, concurrent and distributed systems modeling, database design and reengineering. These and further applications are presented in [5; 6]. Just to mention one application some more precisely, graph grammars are used to define the syntax of visual languages. On the basis of independence results for graph transformation, parsing algorithms for context-sensitive visual languages have been developed (cf. [5]).

The theory of graph transformation has made tremendous progress in the last decade, in particular due to the achievements of several projects funded by the European Union. Currently, the TMR Network GETGRATS where the emphasis is laid on theoretical aspects, and the ESPRIT Working Group APPLIGRAPH, mainly concerned with applications and technological potentials of graph transformation are funded (see [2]).

2. MAIN CONTRIBUTIONS TO THEORETICAL COMPUTER SCIENCE

Graph transformations have been formalized in different ways based on set theory, algebra logics and category theory respectively. In each of those cases a graph transformation approach has been developed which is presented in [4]. The concept of graph transformation systems generalizes on one hand the classical concept of Chomsky Grammars on strings and on the other hand tree resp. term rewriting to graph rewriting. Moreover, essential results from the theory of term rewriting systems have been generalized to graph transformation systems. On the other hand, graph transformation influenced term rewriting by the development of a theory of term graph rewriting. More recently, it has been shown how to generalize Petri nets to graph transformation systems, where net markings and transitions are generalized to graphs and graph rules respectively. Compared with graph transformation, Petri nets do not allow to express relations between tokens. Moreover, context tokens which have to exist but are not consumed by the firing of transitions are not allowed. To meet the second restriction, contextual nets are introduced taking the idea of rewriting context over from graph transformation.

All the basic concepts of concurrency have been studied in the framework of graph transformation systems. In particular, concepts for processes unfolding and an event structure semantics have been developed for graph transformation systems (see [1]) such that graph transformation systems can be considered to be a new promising model of concurrency. Graph transformation has been used as a semantical model for actor systems, concurrent constraint programming and the π-calculus (cf. [6]).

Last but not least the graph transformation approach is on the way to become an important formal specification technique for all kinds of communication based systems, including concurrent, distributed, and reactive systems.

Especially for these kinds of systems formal validation of system properties is an important issue. For graph transformation it is possible to formulate graphical consistency conditions concerning existence and non-existence of subgraphs, cardinality restrictions for the number in- and outgoing edges, etc. Logical formulas based on these constraints can be proven for all derived graphs in a way that they are transformed into equivalent application conditions for rules. A transformation where these application conditions are satisfied preserves consistency of graphs [3].

By now, graph transformation is a well established field in theoretical computer science. New theoretical results are developed along the requirements from application fields, e.g. a loose semantics for graph transformation has been developed, because it is needed in software engineering to formalize view-based system development.

3. FUTURE ROLE

According to the requirements for formal specification techniques in the theory of graph transformation systems different kinds of semantics, structuring and verification techniques are under development. These techniques have to be formally related and compared with those of other formal specification techniques. On the other hand, the theory of graph transformation will be further extended to become even more important as a model of concurrency and distribution. Moreover, graph transformations will play an important role for the formalization of semi-formal graphical specification techniques and all kinds of visual languages. In combination with other specification techniques, graph transformation can be advantageously used to visually specify software systems. Combining graph transformation with temporal logics, graph-based validation of dynamic system properties can be supported. In view of the importance of visual programming and visualization of specifications, graphical representation and graph transformation has a great potential to become a new important programming and specification paradigm.

REFERENCES

[1] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. An Event Structure Semantics for Safe Graph Grammars. In *Proc. PROCOMET'94, IFIP TC2 Working Conf., San Miniato 1994*, pages 417–439. IFIP TCS, 1994.

[2] A. Corradini and H.-J. Kreowski. GETGRATS and APPLIGRAPH: Theory and Applications of Graph Transformation. *EATCS Bulletin*, 63, October 1997.

[3] R. Heckel and A. Wagner. Ensuring Consistency of Conditional Graph Grammars – A constructive Approach. *Proc. of SEGRAGRA'95 "Graph Rewriting and Computation", Electronic Notes of TCS*, 2, 1995. http://www.elsevier.nl/locate/entcs/volume2.html.

[4] G. Rozenberg (ed.). *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations.* World Scientific, 1997.

[5] G. Rozenberg et.al. (eds.). *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 2: Applications, Languages and Tools.* World Scientific, 1999. To appear.

[6] G. Rozenberg et.al. (eds.). *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 3: Concurrency, Parallelism and Distribution.* World Scientific, 1999. To appear.