Workshop on Graph Computation Models

Mohamed Mosbah[1] and Annegret Habel[2]

[1] LaBRI - ENSEIRB - University of Bordeaux 1
351 Cours de la Libération, 33405 Talence, France
mosbah@labri.fr

[2] Fachbereich Informatik, Universität Oldenburg
Postfach 2503, D-26111 Oldenburg, Germany
habel@informatik.uni-oldenburg.de

Abstract. A variety of computation models have been developed using graphs and graph transformations. These include models for sequential, distributed, parallel or mobile computation. A graph may represent, in an abstract way, the underlying structure of a computer system, or it may stand for the computation steps running on such a system. In the former, the computation can be carried on the corresponding graph, implying a simplification of the complexity of the system. The aim of the workshop is to bring together researchers interested in all aspects of computation models based on graphs, and their applications. A particular emphasis will be made for models and tools describing general solutions.

1 Graph computation models

There are a lot computation models based on graphs, and their applications. The computation models include mobile computing, programming, data transformations, concurrent and distributed computing. In the following, we will sketch some of these graph computation models.

1.1 Graph relabeling systems & distributed computing

Graph relabeling systems have been successfully used as a suitable tool for encoding distributed algorithms, for proving their correctness and for understanding their power. In this model, a network is represented by a graph whose vertices denote processors, and edges denote communication links. The local state of a processor (resp. link) is encoded by the label attached to the corresponding vertex (resp. edge). A rule is a local transformation of labels. A relabeling system is defined by a finite set of such rules. The application of the rules are asynchronous: there is no global clock available, and two conflict-free applications of rewriting rules may occur simultaneously, provided they do not attempt to modify the same local context in the host graph. Thus, the behaviour of the network is defined by its initial labeling and the rule base of the associated local rewriting calculus. Problems of interest in distributed computing include node

election, node enumeration, spanning tree construction, termination detection, synchronisation, inter-node agreements, or local recognition of global properties. These studies rely on rule-based local computations on network graphs on the one hand, and the recognition and classification of certain initial network configurations on the other hand. The non-existence of deterministic distributed solutions to certain problems leads to propose also the investigation of probabilistic distributed algorithms, the formulation of which seems rather simple, but their analysis is difficult. An important aspect is the relationship between the three principal paradigms of distributed computing – local computations, message passing, shared memory – and the comparison of their expressive powers. Similar questions arise where those three paradigms are compared with mobile agent systems. See e.g. [1,2,3,4].

1.2 Graph reduction

Pointer manipulation is notoriously dangerous in languages like C where there is nothing to prevent: the creation and dereferencing of dangling pointers; the dereferencing of nil pointers or structural changes that break the assumptions of a program, such as turning a list into a cycle. The goal is to improve the safety of pointer programs by providing (1) means for programmers to specify pointer data structure shapes, and (2) algorithms to check statically whether programs preserve the specified shapes. In [5], these aims are approached as follows. 1. Develop a formal notation for specifying shapes (languages of pointer data structures); that is the main cfoncern of this paper. We show how shapes can be defined by graph reduction specifications, which are the dual of graph grammars in that graphs in a language are reduced to an accepting graph rather than generated from a start graph. Polynomially terminating graph reduction specifications whose languages are closed under reduction allow a simple and efficient membership test for individual structures, yet seem powerful enough to specify all common data structures. 2. The effect of a pointer algorithm on the shape of a data structure is captured by abstracting the algorithm to a graph rewrite system annotated with the intended structure shape at the start, end and intermediate points if needed. A static verifier then checks the shape annotations.

1.3 Term graph rewriting

The theory of term graph rewriting allows to reason about computations on expressions with shared subexpressions. Sharing improves the efficiency of computations in space and time, and is ubiquitous in implementations of functional and logic programming languages, systems for automated deduction, and computer algebra systems. Term graph rewriting provides a model to reason about the correctness, completeness and efficiency of term rewriting with shared subexpressions. This model reflects the properties of real implementations more adequately than the conventional, tree-based model of term rewriting. Sharing makes term graph rewriting different from term rewriting with respect to both

efficiency and properties such as termination and confluence, thus requiring a theory different from established term rewriting theory. See, e.g., [6,7].

2 Organization

The workshop will include tutorials, contributed papers, and system demonstrations. The tutorials will introduce many types of graph transformations and their use to study computation models. The system demonstrations based on graph computation models.will range from alpha-versions to fully developed products that are used in education, research or being prepared for commercialisation.

Workshop chairs and organizers

Mohamed Mosbah	University of Bordeaux 1	France
Annegret Habel	University of Oldenburg	Germany

Program commitee

Frank Drewes	Umea University	Sweden
Rachid Echahed	IMAG, Grenoble	France
Emmanuel Godard	University of Marseille	France
Stefan Gruner	University of Pretoria	South Africa
Annegret Habel	University of Oldenburg	Germany
Dirk Janssens	University of Antwerp	Belgium
Hans-Jörg Kreowski	University of Bremen	Germany
Mohamed Mosbah	University of Bordeaux 1	France
Detlef Plump	University of York	United Kingdom

References

1. Godard, E., Métivier, Y., Muscholl, A.: Characterizations of classes of graphs recognizable by local computations. Theory of Computing Systems **37** (2004) 249–293
2. Derbel, B., Mosbah, M.: Distributed graph traversals by relabelling systems with applications. ENTCS **154**(2) (2006) 79–94
3. Bauderon, M., Métivier, Y., Mosbah, M., Sellami, A.: From local computations to asynchronous message passing systems. Research Report RP 1271-02, Université Bordeaux I (2002)
4. Derbel, B., Mosbah, M., Gruner, S.: Mobile agents for implementing local computations in graphs. In: Graph Transformations (ICGT 2008). LNCS, Springer (2008) This volume.
5. Bakewell, A., Plump, D., Runciman, C.: Specifying pointer structures by graph reduction. Mathematical Structures in Computer Science (2008) To appear.
6. Plump, D.: Term graph rewriting. In: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 2: Applications, Languages and Tools. World Scientific (1999) 3–61
7. Echahed, R.: On term-graph rewrite strategies. ENTCS **204** (2008) 99–110