Graph Transformation Modules
and Their Composition*

Frank Drewes, Peter Knirsch, Hans-Jörg Kreowski, and Sabine Kuske

Department of Computer Science
University of Bremen
P.O. Box 33 04 40
D-28334 Bremen (Germany)
{drewes, knirsch, kreowski, kuske}@informatik.uni-bremen.de

Abstract. In this paper, we investigate the notion of transformation modules as a structuring principle for the specification of graph transformation systems which provide a collection of operations on graphs. Based on the notion of transformation units, a concept that allows to specify binary relations on graphs, a transformation module consists of a set of transformation units. To be able to distinguish between hidden and public operations, a module has an export interface. Moreover, there may be an import interface and a formal parameter. The import interface allows the use of transformation units which are known in the environment of a module. The formal parameter consists of formal parameter units which specify operations on graphs in a loose way. A formal parameter unit may be instantiated by an exported transformation unit of another module through module composition.

1 Introduction

Graph transformation is a rule-based framework for the modeling and analysis of information-processing systems the states of which are represented by graphs in a natural way. Although one encounters a number of applications of this kind in the literature (see, e.g., the Handbook of Graph Grammars and Computing by Graph Transformation and its Vol. 2 in particular [Roz97, EKR99, EKMR99]), graph transformation is not yet used frequently in practice. However, there is some hope that this may change in the future and that graph transformation will be recognized as a useful and adequate specification and programming approach (see, e.g., Andries et al. [AEH+99] for a more detailed discussion). The perspectives depend on many factors one of which is the quality and availability of structuring principles.

In this paper, we contribute to the topic of structuring and study the notion of transformation modules and their composition which allow the modular development of large graph transformation systems from small entities. As a transformation module is intended to provide a collection of operations on graphs and a

* Partially supported by the EC TMR Network GETGRATS (General Theory of Graph Transformation Systems) and the ESPRIT Working Group APPLIGRAPH through the University of Bremen.
transformation unit is a proven concept for the specification of a single operation on graphs (see [KK96, KKS97, AEH+99, KK99]), a transformation module encapsulates a set of transformation units. To be able to distinguish between hidden and public operations, a module has an export interface. Moreover, there may be an import interface and a formal parameter. The import interface allows the use of transformation units which are known in the environment of a module. The formal parameter consists of formal parameter units which specify operations on graphs in a loose way. A formal parameter unit may be instantiated by an exported transformation unit of another module through module composition. It turns out that the result of a sequence of compositions depends only on the assignments of actual export units to formal parameter units, and not on the order in which successive modules are composed. In other words, it is meaningful to develop a graph transformation system as a network of small transformation modules which are connected by parameter assignments.

To avoid unnecessary technicalities, we consider operations on graphs as binary relations, i.e. as input-output relations between initial and terminal graphs, because this is the most intuitive and straightforward interpretation of rules and rule applications in a rule-based framework. However, operations with arbitrary arity on various types are implicitly available because many data structures have nice graphical representations and a number of graphs can be seen as a single graph through the disjoint union.

The paper is organized in the following way. The central concept of transformation modules is introduced and discussed in Section 4 based on transformation units which are recalled in Section 2 and on formal parameter units which are introduced in Section 3 as a counterpart to transformation units with loose semantics. In Section 5, we define the composition of modules which allows the instantiation of formal parameter units of one module by export units of another module. All concepts are illustrated by a running example based on Dijkstra's shortest paths algorithm. In Section 6, we discuss the relation of transformation modules to other module concepts in the framework of graph transformation. In contrast to the notions in [TS95, SW98, GRPS99] which are based on particular graph transformation approaches, transformation modules are approach independent, i.e. independent of a particular choice of graphs, rules, and direct derivations. Actually, our module concept follows the lines of research that led to the notion of transformation units and simple modules as structuring concepts of the graph and rule centered language GRACE (see [KK96, AEH+99, KKS97, HHHK98, KK99]). Apart from some minor differences, transformation modules generalize simple modules in the sense that they additionally contain formal parameters with a loose semantics. In the conclusion, we briefly summarize the presented concepts and outline further research topics.

2 Transformation Units

In this section, we recall the concept of transformation units and their inter-leaving semantics. Transformation units provide a structuring concept on the