On the Analysis and Verification of Graph Transformation Systems

Adel Torkaman Rahmani, Vahid Rafe
Department of Computer Engineering
Iran University of Science and Technology
Tehran, Iran
{rahmani, rafe}@iust.ac.ir

***Abstract*—Graph Transformation has recently become more and more popular as a general formal modeling language. Easy to use by designers and a considerable capability to behavioral modeling of dynamic systems characterize it among other formalisms. In this paper we concentrate on how to analyze these models. We will describe our approach to show how one can verify the designed graph transformation systems. To verify graph transformation systems we use a novel approach: using Bogor model checker to verify graph transformation systems.**

I. INTRODUCTION

Most of the artifacts software engineers are dealing with are nothing but suitable annotated graphs. Software architectures, class diagrams and version histories are only a few well-known examples in which graphs have proven their usefulness in everyday software engineering. These models, and many others, can easily be described by means of suitable graph transformation systems [1]: They can help formalize the syntax behind these models, but they can also be useful to define the formal semantics of these notations [2, 3].

Rule based nature of graph transformation systems can play an important role in modeling of complex and large systems. But so far, most of the research concentrated on graph transformation systems as a modeling means, without considering the need for suitable analysis tools. Oftentimes, modeling is not enough since users want to be able to "discover" the interesting properties behind their models. This is why even the *perfect* graph transformation system must be complemented with automated analysis capabilities to let users reason on it and understand whether the transformation system fulfills their requirements. And model checking has proven to be a viable solution for this purpose.

The two most relevant approaches that merge graph transformation systems and model checking are CheckVML [4, 5] and GROOVE [6]. Despite their powerful theoretical background, their usability for model checking complex graph transformation systems is limited. We will explain these limitations in the next section.

The proposal presented in this paper addresses these limitations and adopts Bogor [7] for the model checking of complex graph transformation systems. In our approach, graphs are translated into BIR (Bandera Intermediate Language [8] -the input language of Bogor- while properties are rendered by means of LTL (Linear Temporal Logic) and special-purpose rules. The result is used to feed Bogor that performs the verification.

The paper is organized as follows. Section 2 surveys the state of the art. Section 3 briefly introduces Bogor and motivates the choice of this model checker with respect to other options. In section 4, we briefly introduce attributed typed graph transformation systems. Section 5, describes our approach and shows how we encode a graph transformation system in BIR and presents some experimental results and compares them with existing approaches. Section 6 concludes the paper.

II. RELATED WORK

The theoretical foundations for the verification of graph transformation systems through model checking have been studied thoroughly by Heckel et al. in [9]: graphs are interpreted as states and transformation rules as transitions. This idea is exploited by both GROOVE [6] and CheckVML [4], as well as by our approach.

GROOVE applies graph-specific model checking algorithms by rendering graphs as states and transitions as application of graph transformation rules. Properties are specified as transformation rules and CTL expressions containing rule names as atoms. Since GROOVE does not support typed graphs, the verification of real models becomes complex (or unfeasible). There is a proposal to extend GROOVE with attributed graphs [10], but it still does not support all the "common" types (e.g., strings) and is complex and difficult for the designers since attributes are separated from their values. Since real models can have nodes with many attributes, GROOVE graphs are not easy to understand, and its performance deteriorates with respect to the size of the graph.