Simulating Algebraic High-Level Nets by Parallel Attributed Graph Transformation

Claudia Ermel[1], Gabriele Taentzer[1], and Roswitha Bardohl[2]

[1] Technische Universität Berlin, Germany
{lieske,gabi}@cs.tu-berlin.de
[2] Intern. Center for Computer Science, Schloss Dagstuhl, Germany
rosi@dagstuhl.de

Abstract. The "classical" approach to represent Petri nets by graph transformation systems is to translate each transition of a specific Petri net to a graph rule (behavior rule). This translation depends on a concrete model and may yield large graph transformation systems as the number of rules depends directly on the number of transitions in the net. Hence, the aim of this paper is to define the behavior of Algebraic High-Level nets, a high-level Petri net variant, by a parallel, typed, attributed graph transformation system. Such a general parallel transformation system for AHL nets replaces the translation of transitions of specific AHL nets. After reviewing the formal definitions of AHL nets and parallel attributed graph transformation, we formalize the classical translation from AHL nets to graph transformation systems and prove the correctness of the translation. The translation approach then is contrasted to a definition for AHL net behavior based on parallel graph transformation. We show that the resulting amalgamated rules correspond to the behavior rules from the classical translation approach.

1 Introduction

Visual modeling languages (like the Unified Modeling Language (UML), Petri nets, Statecharts, and many more) play a central role for software and system modeling. Visual models are used for system design, simulation, validation, and code generation. Apart from developing visual models, the simulation of a model on the basis of a formal specification is an important issue for testing and validating the system behavior. The simulation of Petri nets, for example, is realized by playing the token game: a transition can fire if it is enabled, a firing step removes tokens from the transition's predomain places and adds tokens to its postdomain places.

Petri net behavior can be defined as graph transformation system where each transition is translated to a graph rule modeling the corresponding change of the marking (deleting and/or adding tokens) in a firing step [15, 3]. This "classical" way to define Petri net behavior by graph transformation assumes a specific Petri net before compiling its transitions into graph rules (*compiler* approach).

Yet, for related visual behavior modeling languages, it is often possible to define a general graph transformation system which is independent of a specific

H.-J. Kreowski et al. (Eds.): Formal Methods (Ehrig Festschrift), LNCS 3393, pp. 64–83, 2005.

model and can be used to interpret arbitrary models of a visual language (*interpreter* approach). An example is a graph transformation system for describing the behavior of a Statechart variant given in [2]. In general, the interpreter approach is much more flexible and scalable than the compiler approach. As it is independent of a concrete model, the graph transformation system defined for the interpreter approach is fixed once for the complete visual language, i.e. the number of behavior rules is finite and does not grow with the size of the model (scalability). In contrast, using the compiler approach, each specific model must be translated to get the model-specific graph transformation system.

Unfortunately, it is difficult to give a general graph transformation system to simulate Petri nets as there may be arbitrary many places connected to a transition, leading to an arbitrary number of behavior rules. Hence, parallel graph transformation concepts have been used to simulate the behavior of Condition-Event nets in [20] and of *Timed Transition Petri Nets* in [4].

Parallel graph transformation was introduced by Ehrig and Kreowski in [6], later generalized to parallel high-level replacement systems [11] by Ehrig and Taentzer, further elaborated and applied to communication-based systems in [20]. The essence of parallel graph transformation is that (possibly infinite) sets of rules which have a certain regularity, so-called rule schemes, can be described by a finite set of rules modeling the elementary actions. For instance, when modeling the firing of a Petri net transition, the elementary actions would be the removal of a token from a place in the transition's predomain and the addition of a token to a postdomain place. For the description of such rule schemes the concept of amalgamating rules at subrules is used which is based on synchronization mechanisms for rules developed first in [5].

The aim of this paper is to present a formal interpreter approach to define the behavior of high-level Petri nets. A specific, well-defined variant of high-level nets are Algebraic High-Level nets, AHL nets for short, introduced by Ehrig, Padberg and Ribeiro in [18]. We present an interpreter approach for the behavior of AHL nets based on parallel attributed graph transformation. Thus, a general graph transformation system for simulating AHL nets replaces the translation of transitions of specific AHL nets. The resulting parallel behavior specification is formally proven to be semantically equivalent to the corresponding compiler approach translating each specific AHL net to a corresponding attributed graph transformation system. This compiler approach for AHL nets has been presented in [1] and is reviewed in a slightly modified form in this paper.

In Section 2, the formal definitions of AHL nets and their behavior are reviewed, using the well-known *Dining Philosophers* as running example. Section 3 presents the concepts of sequential (classical) and parallel attributed graph transformation. The concepts are the basis in Section 4 to formalize the translation from AHL nets to sequential graph transformation systems according to the compiler approach. We prove the semantical compatibility of an AHL net and its translation to a graph transformation system, i.e. we show that a firing sequence in the net corresponds to a graph transformation sequence in the translated graph transformation system. The compiler approach is contrasted by