Mapping UML Models to Colored Petri Nets Models based on Edged Graph Grammar

Zhanwei DU, Yongjian YANG †, Jiaqi XU, Jie WANG

College of Computer Science and Technology, Jilin University, Changchun 130012, China

Abstract

This paper presents an approach for transforming UML class diagrams and extended sequence diagrams into Colored Petri Nets. In the field of software model, UML has become a widely used software modeling language. However, as a semi-formal language, UML lacks a precise formal semantic base for its notation. In this paper, Colored Petri Nets are taken as the formal language for UML models' analysis. Intermediate graphs generate UML extended class diagrams and extended sequence diagrams. Edged Graph Grammar is introduced to describe the transformation rules. In order to show the approach's process, an case study is shown.

Keywords: Colored Petri Nets (CPN); UML; Models Transformation; Edged Graph Grammar

1. Introduction

Unified Modeling Language (UML) is considered nowadays as the industry standard as a common object-oriented modeling language. Plenty of advanced concepts of software engineering have be added to UML. But, UML lacks powerful semantic base for its own analysis, thus it makes UML hard for its notation.

Petri nets and in particular Colored Petri nets have not only an intuitive graphical representation, but also strict mathematical definition which is appropriate for systems modeling and analysis. Through transforming UML models to Colored Petri nets for analysis, the results of the formal analysis are back-annotated to the UML models to hide the mathematics from modelers of UML. It is significant for the modelers to find defects in the design stage. And how to do the transformation rationally and effectively has become the focus of software engineers and Petri theorists.


† Corresponding author.
Email addresses: yyj@jlu.edu.cn (Yongjian YANG).
Those transforms aim to bridge the gap between UML models for analysis purpose and have tried to let the automatic generation of a tool check and build models according to the formalism's syntax. However, the computational complexity of these algorithms is high due to embedding, inference and induction problem.

In order to overcome these existing problems, we introduce EGG[7] which is polynomial complexity as a context-sensitive graph grammar. It uses the graph edges to represent the context information, as well as its simple rules to solve the problem of dangling edges, besides its production's structure is simple.

The UML Class diagrams are the static structure diagrams that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. It suggests the color set of Petri nets. The UML sequence diagrams are the interaction diagrams that emphasized the structural organization of the objects which send and receive messages. To keep constraints, the extended Class diagrams are still introduced here. And to express the concepts of synchronism and asynchronism, we use the extended sequence diagrams[8]. In this paper, UML Class diagrams and sequence diagrams are combined to make intermediate diagrams. Based on the intermediate diagrams, we transform the UML models into Colored Petri nets successfully. With this end, a case study is shown. It reveals the validity of our approach.

The remainder of the paper is organized as follows: section 2 outlines the transformation model. In Section 3, we illustrate our work with a case study. Finally, section 4 concludes the paper.

2. The Transformation Model

Our approach consists of a process with two steps. First, UML Class diagrams and sequence diagrams are combined into equivalent intermediate diagrams. In the second step, the obtained intermediate diagrams are used to generate the equivalent Colored Petri nets for further analysis. The approach is illustrated in Figure 1.

![Fig.1 Architecture of the Approach](image)

2.1. The Intermediate Diagram

In the first step of our approach, the extended sequence diagrams are used in order to express the concepts of synchronism and asynchronism. There are many kinds of extended sequence diagrams. Here, we just use the extended sequence diagrams propose by Wang[8], for its excellent user-friendly interface. As for UML Class diagrams, there are still many kinds. Here, we just use the UML Class diagrams proposed by Zhou[9] for its association.
A class has either attributes or methods. And to express constraints, another box is added to the class diagram as the fourth. As shown in Figure 2, the class diagram has four boxes, which represent class's name, properties, methods and constraints.

To transform a Sequence diagram and Class diagrams into a intermediate diagram, we have proposed 9 rules which are shown in Figure 3.

Here, we do not need to study in-depth from the language point of view, so we just need to give the production and transformation rules with the graph rewriting system EGG (Graph Rewriting) describing the rules.

The idea of the transformation can be summarized on the following main steps: The first step is to convert every object of Sequence diagram to the extended class shown as in Figure 2. The second step is to map the graph obtained to the final intermediate graph according to the rules shown in Figure 3.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig2.png}
\caption{Extended Class}
\end{figure}

2.2. The Proposed Approach

Once the intermediate graphs are generated, we can perform their mapping to the equivalent CPN models for analysis and simulation.

EGG's production is shown in Figure 4, these symbols are all terminal symbols except extended class as nonterminal symbols.

The method of setting the color set

1.1 Define the different attributes of the class as the Atomic color sets, each color set could have the same name as its corresponding attribute.

1.2 Define a class color set as the production of its attributes' atomic color sets.

1.3 Define variables for each atomic color sets, and still define variables for classes if necessary.

3. Case Study

To illustrate our approach, we consider an example of login which has been described by the UML Class diagrams (figure 5) and the UML Collaboration diagram(figure 6). User click the link of login on the site page, then system displays the login page. After that, user enter his name and password in the login form and click the submit button. Until system validate user's information, the login page is displayed. And the declarations are shown in Figure 7.

In order to analyze the UML model, we have to transform the UML Class diagrams and the UML sequence diagrams to their extended forms. After that we could obtain the intermediate graph. The resulted Colored Petri nets is shown in Figure 7 and Figure 8. We can now use any CPN tool to analyze the resulted model.
Fig. 3 The Transformation for Intermediate Diagram

Fig. 4 The Transformation for CPN
Fig. 5 Class Diagrams

Fig. 6 Extended UML Collaboration Diagram of Login
Fig. 7 Colored Petri Nets I

Fig. 8 Colored Petri Nets II
4. Conclusion and Future Work

In this paper we proposed an approach for mapping UML extended Class diagrams and sequence diagrams to Colored Petri nets models. This transformation aimed to bridge the gap between informal notation and more formal notation for analysis purpose. The approach is based on edged graph transformation since the input and output of the transformation are graphs. An example illustrates our approach.

In a future work we plan to combine the meta model into the approach in order to put into practice.

Acknowledgement

This work was supported by Graduate Innovation Fund of Jilin University(20111064), Jilin province science and technology development plan item (20080319), Jilin Province Development and Innovation Committee’s High and New Technology Projects(20106421) and Jilin Province Key Scientific and Technological Projects(20100309).

References