A Research of Model Checking UML Statechart Diagrams

Lin Hsin-Hung(110205)

School of Information Science,
Japan Advanced Institute of Science and Technology

August 15, 2003

Keywords: model checking, SMV, UML, statechart.

1 Introduction

The United Modeling Language(UML) has become a standardized notation for specifying complicated software systems. By the help of UML, the software design becomes more and more complicated so that the integrity of a software system design is very difficult to be discovered. Further more, because UML itself is a very expressive and rich language, sometimes the model gives behaviors not expected by the designers and those behaviors could cause serious bugs for the system. For this reason, verifications about UML models are needed.

Recently there are some researches using Model Checking technique for verification of UML models. Model checking is an automatic technique for verifying finite state concurrent systems. It has been applied successfully to hardware design such as verifying complex sequential circuit designs and communication protocols. Recently there are many researches focus on using model checking techniques on software verification. Because UML statechart diagrams and automata, which can be easily converted to Kripke structure, have many similarities, it is an interesting subject to apply model-checking techniques on Statechart diagrams.

2 Purpose and Approach

In this research, we attempt to describe an approach to convert UML model into SMV module. We will focus on translating statechart diagrams, and the purpose of this research is to explore the states in a UML statechart diagram and verify properties interested.

Our approach will focus on constructing an algorithm to translate UML statecharts into SMV. We will follow the STP-approach since it successfully translates STATEMATE statecharts into SMV. Unfortunately, STP-approach could not handle the message passing between UML statecharts; because STP considers STATEMATE statechart a closed system and all events are unique in the system. For this reason, we need to adopt some parts of the STP algorithm for message passing of UML statecharts.

The message passing consideration refers to the algorithm introduced in HUGO. But differently, we do not use queues for messages but only applied the similar concept in our translation into SMV. This is because the algorithm of HUGO is for translation into PROMELA, a C-like language, such that the implementation of a queue is much easier than in SMV.

To construct the relationship between UML statecharts, we need to take a reference to class diagrams. In this way we might not have to handle events as global boolean variables as in STP-approach. In this research we cannot afford to construct a mechanism of making specification properties.

3 Translation Algorithm

The translation algorithm contains the following sections:

- **State/statechart and sub-state/statechart**
- **State transitions**
- **Event variables**
- **Mutual exclusive message passing**

The first two sections are based on STP-approach and we made a little adaption to match the structure of UML statechart diagrams. The last two sections are concerning about dynamic mechanism of UML statechart diagrams. We made our own explanation about the message passing since the UML semantics does not give clear definition.

The mechanism of mutual exclusive message passing restricts that a statechart can only accept message passing from one statechart (including from itself) at one time step. When a message passing clash occurs, only one message passing is allowed and others are forced to delay till next time step.

4 Examples and Conclusion

We conducted an example of Dining Philosophers Problem that has two philosophers and two forks. While translated by our algorithm, we then examinated some properties interested and concerning about dead lock of thie problem. The results shows that our translation algorithm works well in this example.

There is still work could be continued form this algorithm such as making its coverage about UML statechart diagrams more complete, and implement an automatic translating tool. While testing our algorithm with the example, we also found that making a proper specification property is somehow difficult. This is also as important as translating correctly into SMV code.