# A Study on Transformation from Timed Automata to Timed Modules

Yoshinobu Tachi

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

February 15, 2001

**Keywords:** timed automata, timed modules, real-time systems, software process.

## 1 Background

For real-time systems, such as communication systems, logic circuits, operating systems etc., we must ensure not only that they work correctly but also that they work in correct timing. Recently, many real-time systems are very large and complex, so that it is very difficult to verify correctness of them by simulation tests. Hence, we need mathematical verification methods in the design step of them.

For this purpose, that is, for verification of correctness of real-time systems, timed automata and timed modules are proposed. The timed automata model is an extension of a well-known model, automata. Each timed automaton has clocks to control its transitions: each edge of a timed automaton may have clock constraints and clock resets. The timed module model is the model which describes systems by transitions of the values of variables. The types of variables are discrete(boolean or enumerate) and clock(real). That is, by using this model, we can treat both actions concerning discrete variables and action concerning clocks by the same manner. Both of the two models are proposed by Alur et. al. They suggests that the timed module model is a generalization of the timed automata model. However, they did not explicitly give any transformation procedure from timed automata to timed modules.

# 2 Transformation from Timed Automata to Timed Modules

Then, in this thesis, we give a transformation procedure from timed automata to timed modules. The procedure consists of the following two sub-procedures.

- An equivalent transformation procedure from timed automata to timed modules
- An optimization procedure for the timed modules obtained by the above procedure

We also give the proofs of the validities of them. Note that the procedure is for the timed I/O automata model. This is the model with Input/Output distinguish in the events. We use this model because systems and processes that we want to describe need such events. We apply this procedure to the following two examples.

- A specification of a train/gate controller system
  (This is an example of real-time systems.)
- "Software Process Modeling Example Problem" by Kellner et. al. [3]

It is one feature of this thesis that we give a software process description in an automata model. We have compared the description obtained by the procedure with the description in timed automata, for both examples. We conclude that the descriptions obtained by applying the procedure are certainly optimized. This is due to discrete variables of timed modules, which give us flexible descriptions. Details are shown in the next section.

# 3 Comparison between Two Models

Because only one event are allowed to occur for each transition in timed automata, we need to write descriptions which are not simple, in the following cases.

- For the case that an event can occur only when some other input events have arrived, we need the same number of edges and states as the number of the input events.
- It is similar for the case that an event can occur only after some output events have occurred.
- For each of the above cases, if the order of the input(output) events can not be determined, then the number of the possible orders is the factorial of the events, and we need the same number of states and edges as the number of the orders.
- For an input event and a continuously succeeding output event, we also need two edges and states.
- Suppose that we need a judgment in a description. We need the same number of states and edges as the number of the results of the judgment.

- Suppose that each timed automaton must let its state be the initial state by some "reset event" of another automaton. We need the same number o9f additional edges as the number of the states of the automaton.

By using timed modules, we can write simpler (optimized) descriptions for each of the above cases.

- In timed modules, each input event is expressed by the assignment of "true" for the boolean variable corresponding to the event, as a condition of an expression. Hence, we can describe the arrivals of many input events by the conditions of only one expression.
- It is similar for the case of output events.
- In timed modules, the order of events becomes the order of the conditions of an expression. Because we need not consider this order, we need only one expression also for this case.
- An input event becomes a condition of an expression and a continuously succeeding output event becomes an assignment in the expression. Hence, we need only one expression also for this case.
- In timed modules, the results of a judgment are expressed by the values of a variable with enumerate type. Hence, we need only one variable for any judgment.
- For a "reset event", we must add only one expression which lets the value of the variable "state" be the value denoting the initial state whenever the current value of the "state" is.

In the examples in the chapter 4, we have shown the optimized descriptions in timed modules obtained by applying the above methods. Note that descriptions in timed modules are sets of expressions, so that they are not easy to read. Hence, we also propose an "automata representation" of timed modules, in this thesis.

# 4 Future Works

Future works are as follows.

1. Application of the transformation procedure to other examples, that is, other examples of real-time systems and other software processes.
2. Development of further optimization methods.