A Graph Grammar Approach for Durational Action Timed Automata Determinization

Djamel Eddine Saidouni, Ilhem Kitouni, Hiba Hachichi

MISC Laboratory, Computer Science Department, Faculty of Engineering, University Mentouri Constantine, 25000, Algeria
Saidounid@hotmail.com, ilhkit2@yahoo.fr, hiba.hachichi@gmail.com

ABSTRACT

Durational Action Timed Automata (DATA) is a semantic model for expressing the behavior of real time systems where actions have durations. In this paper, we propose an approach for translating a DATA structure to a corresponding deterministic one. For this purpose, a meta-model of DATA model and a transformation grammar are defined. Programs are written in Python language and implemented under the ATOM³ environment.

Keywords: Concurrent system, Formal Verification, Graph transformation, DATA, Determinization, ATOM³, Meta-models, Formal testing approach.

1. INTRODUCTION

Nowadays, technology is looking for distributed applications to develop and increase its domains (network, telecommunication...etc). This kind of applications is known by their big complexity. Formal verification method is the most used technique to deal with concurrent systems questions because of its ability to describe the system behavior without ambiguity; it offers several verification approaches for assessing systems behaviors. These approaches are based on semantic models. In this paper we are interested by formal testing approach of real time systems in which actions may elapse in time. The durational action timed automata will be used as a semantic model of such systems [3, 4]. Because formal testing approach is based on a deterministic structure, in this paper we propose an approach for translating a DATA structure to a corresponding deterministic one [9]. For this purpose, a meta model of DATA model and a transformation grammar are defined. Programs are written in Python language and implemented under the ATOM³ environment [1,10].

The paper is organized as follows. In section 2, we recall some basic definitions about Durational Action Timed Automata. Model transformation approach is presented [2,5,13,14], especially graph transformation methods and related tools particularly ATOM³. Section 3 describes our approach of transforming DATA structures to corresponding deterministic ones. The proposed approach is based on graph grammars. Section 4 illustrates the approach through an example. The final section concludes the paper and gives some perspectives.

2. BACKGROUND

Our objective is to propose an automatic generation of deterministic DATA [04,18] with a lot of states using graph transformation method. In the following, we recall some basic definitions about Durational Action Timed Automata[15,16,19], determinization process and graph transformation.

2.1 DURATIONAL ACTION TIMED AUTOMATON “DATA’S MODEL”

To illustrate this Model, we consider the example of system S (FIGURE 1). From the initial state $s_0$, the actions $a$ and $b$ can comply independently, in particular at the same time. Clocks identifiers $x$ and $y$ are assigned to actions $a$ and $b$ respectively. Therefore, starting from state $s_0$, the two following transitions are possible: $s_0 \rightarrow s_1$ and $s_0 \rightarrow s_2$.

A transition labeled with $a$ indicates the beginning of execution of the action $a$, the associated clock counts the evolution in the time of this action. Following the same reasoning, the two following transitions are possible: $s_1 \rightarrow s_3$ and $s_2 \rightarrow s_3$.

The behavior of the system $S$ is given by Figure 1.(a). Starting from state $s_3$, the action $d$ can comply only if the two actions $a$ and $b$ finished their execution. Therefore, the transition $d$ can be drawn only if a condition involving to the executions of $a$ and $b$ is satisfied. This condition, called duration condition, is built according to the durations of $a$ and $b$. Initially, we show the construction of duration conditions for $s_0$, $s_1$ and $s_2$. After launching the transition $s_0 \rightarrow s_1$, we need information on the possible execution of the action $a$ in state $s_1$. The termination of action $a$ is expressed by the value of the clock $x$, $x$ is greater than 10 because the duration of action $a$ is equal to 10. Therefore the duration condition \{$x\geq10$\} is added to the state $s_1$. The state $s_2$ is labeled by the duration condition \{$y\geq12$\} because the duration of action $b$ is equal to...
In state $s_0$, no action is complying, which implies that duration condition set is empty. In the state $s_3$, the termination of actions $a$ and $b$ is expressed by the set of duration conditions $\{x \geq 10, y \geq 12\}$. The execution condition of the action $d$ becomes $x \geq 10 \land y \geq 12$.

**Definition 2.3**: For associates to each $d$ over $H$, the termination of actions potentially complying in $s$ becomes $x, y := 0$.

**Definition 2.4** [9]: The semantics of a DATA $A = (S, L, s_0, H, T)$ is defined by associating to it an infinite transition system $S_A$ over $Act \times R^*$. A state of $S_A$ (or configuration) is a pair $<s, v>$ such as $s$ is a state of $A$ and $v$ is a valuation for $H$. A configuration $<s_0, v_0>$ is initial if $s_0$ is the initial state of $A$ and $\forall x \in H$, $v_0(x) = 0$. Two types of transitions between $S_A$ configurations are possible, and which correspond respectively to time passing (rule RA) and the launching of a transition from $A$ (rule RD).

For more details, the reader is referred to [3, 4].

**2.1.1 Determinization of Durational Action Timed Automata** [9]

The determinization [20, 21, 22, 17] of a DATA is realized according to the following steps:

- The elimination of internal actions ($\perp$ actions).
- The elimination of non-determinism caused by the observable actions.
- The treatment of the divergence.

**2.2 Graph Transformation**

The transformation between models is a process that converts a model to another model. This task requires a set of rules that define how the source model has to be analyzed and transformed into other elements of the target model. The transformation engine takes the source model in input; execute the rules of transformation and finally generate the target model in output.

Graph Grammars [6, 7, 8, 12, 14] are used for model transformation. They are composed of production rules; each having graphs in their left and right hand sides (LHS and RHS). Rules are compared with an input graph called host graph. If a matching is found between the LHS of a rule and a subgraph in the host graph, then the rule can be applied and the matching subgraph of the host graph is replaced by the RHS of the rule. Furthermore, rules may also have a condition that must be satisfied in order for the rule to be applied, as well as actions to be performed when the rule is executed. A rewriting system iteratively applies matching rules in the grammar to the host graph, until no more rules are applicable. ATOM is a graph transformation tool among others. In this paper we use ATOM.
Example of grammar rule in ATOM

FIGURE 3: A grammar rule (LHS and RHS) that eliminate a cycle

In the next section, we will discuss how we use ATOM to generate deterministic DATA models using graph transformation based on ATOM tool.

3. THE APPROACH

In order to allow the consideration of DATA structures of a high number of states and transitions generated we propose firstly a program written in Python language that transforms a DATA structure, presented as a text file according to a particular grammar, to a DATA structure writing in the form of a python file respecting the syntax of ATOM. Also we define a meta-model of the DATA model and a transformation grammar. The meta-model is represented by UML class diagrams and the constraints are expressed using Python language.

3.1 GENERATION OF A DATA RESPECTING THE SYNTAX OF ATOM

Figures 4.a and 4.b present an example of textual and graphical representations of a DATA structure. Figure 5 gives the python file of the DATA structure. The translation from a textual representation to a python representation is done by the python program pgm_DATAfichText_DATAfichAtom3.py.

FIGURE 4.a: A textual representation of a DATA

FIGURE 4.b: A python representation of a DATA seen under the ATOM environment

FIGURE 5: Steps of generating a DATA

3.2 META-MODEL

The meta-model in ATOM is an UML class diagrams and the constraints are expressed in python language [11].
3.2.1 DATA META-MODEL AND A TOOL FOR DATA MODELS

A DATA consists of states (with an initial state) and transitions. So, our meta-model of DATA is composed mainly of two classes (DATAStrate, DATAIInitState) and an association (DATATransition) as shown in figure 6 and described below:

DATAStrate: this class represents the DATA states. Every state has 2 attributes: a name (name) and duration conditions (conDuree). This class is connected to DATAIInitState by inheritance link.

DATATransition: this class describes DATA transitions. Every transition is identified by an Action, clock and the execution conditions (condExecution).

DATAIInitState: this class represents the initial state of the DATA. It inherits its attributes from DATAStrate class. Each class has a unique graphical appearance.

3.2.2 A TOOL FOR DATA MODELS

Based on the meta-model of figure 6, ATOM allows the generation of a tool for DATA models as shown in the tool bar of figure 7.

FIGURE 6: DATA meta-model

3.3 GRAPH GRAMMAR

The proposed graph grammar is composed by 10 rules organized in 4 categories.

- Rules 1 and 2 of figure 8 requiring the system to enter an observable action from the initial state. If the first action is internal (¿ action) the system displays an error message and it blocks.
  - RQ: the internal action is represented with “T”.

- Rules 3 and 4 of figure 9 allow eliminating a “loop of ¿ actions”
  - Rule 3: generates a duration condition which is the sum of all durations of ¿ actions.
  - Rule4: allows generating a precedent duration condition (rule3) provided with the number of iterations represented by ”W”. And replacement of this duration condition in the execution condition of the next transition if it is conditioned by ¿ cycle.
**FIGURE 9:** Eliminate the ↓ cycles

- Rules 5, 6 and 7 of figure 10 allow the elimination of ↓ transitions “sequence of ↓ actions”
  - *Rule 5:* eliminates ↓ transition in case of branches.
  - *Rule 6:* eliminates ↓ transition with successor.
  - *Rule 7:* eliminates ↓ transition without successor.

In rules 5, 6 and 7 we merge the state which precedes the transition and the one which succeeds it in a new state. In this state we keep the same set of duration conditions with a substitution of the duration condition relative to the clock of the internal action “↓” by the execution condition of ↓ transition increased with the duration of ↓.

In rule 5 and 6 will be the same process in execution condition of the next transition, if it is conditioned by ↓ transition.

**FIGURE 10:** Elimination of ↓ transitions

- Rule 8 of figure 11 eliminates unreachable states

**FIGURE 11:** Eliminate unreachable states

- Rules 9 and 10 of figure 12 allow saving a deterministic DATA in a file.

**FIGURE 12:** Save a deterministic DATA

4. Example

We propose this example to illustrate our approach. The mapping of the DATA file of figure 13 to the equivalent DATA model of figure 14 is performed using python program. We have applied our tool on the DATA model and obtained automatically the deterministic one of figure 15. The result is saved in the file of figure 16.
5. CONCLUSION

In this paper, we have proposed an approach and a tool for translating DATA structures to a corresponding deterministic ones using graph grammar since DATA structures are graphs. To perform this transformation, we have proposed a program written in Python language that transforms DATA written in the form of a text file to DATA written in the form of python file respecting the syntax of ATOM³. We have proposed also a meta-model for the DATA model. Based on this meta-model, we have proposed a graph grammar that deals with the transformation process. The meta-modeling tool ATOM³ is used for this purpose. We have illustrated our approach through an example. In future work, we plan to implement a tool which generates the DATA models from D-LOTOS specification. The transformation of deterministic DATA to the testing model “Timed Refusal Graph”[9,23] is in perspective too.
REFERENCES


[22] A Petit V Diekert, P Gastin. Removing e-transitions in timed automata. proceeding of
the 14th annual symposium on theoretical aspect of computer science (STACS.97) number 1200, 1997. lecture notes in computer science pages 583-594.