A Graph Grammar Approach for Calculation of Aggregate Regions Automata

Hiba Hachichi
Ilham Kitouni
Djamel-Eddine Saidouni
A Graph Grammar Approach for Calculation of Aggregate Regions Automata

Hiba HACHICHI, Ilham KITOUNI and Djamel-Eddine SAIDOUNI
MISC Laboratory, University Mentouri Constantine, 25000, Algeria
hiba.hachichi@gmail.com, ilhkit2@yahoo.fr, saidounid@hotmail.com

Abstract:
This paper proposes an approach for transforming DATA* in aggregate region automaton. This study uses the approach of graph transformation and proposes two meta-models associated to DATA* and aggregate regions automata. We propose also a graph grammar for the execution of the transformation; using the modeling tool AToM3.

Keywords: Formal Verification, Graph Transformation, DATA*, Regions Automata, Aggregate Regions Automata, AToM3.

1. INTRODUCTION

Today most computer systems are constructed to apprehend the temporal aspect. This aspect is required by the tasks currently assigned to the computer systems, control tasks and communication tasks. This unavoidable characteristic makes the production of systems more complex, and more prone to errors.

Indeed systems validation (verification or test) always take a special interest among the research areas in computer [1] [7] [18]. It should be noted that the success of these operations of validation depends on the models quality used for analysis, conception and feasibility of different phases. For example, the automation of validation reduces considerably the errors and enables the implementation even if it remains very expensive.

In this context the model of Durational action timed automata (DATA*) arouses more interest particularly for its use in the validation of systems. DATA* is a temporal model, its semantic express the durations of actions and notions for the specification of the real-time systems as urgency and deadline [5] [6]. This model [4] [11] [15] is following of work initiated in [14]. It based on a maximality semantic and that advocates the true parallelism; from this point of view it is well suitable for modeling real time, concurrent and distributed systems.

The regions automata are the automata which reproduce the infinite execution of timed automata by a finite set of transitions. Also it is well known that in the domain of verification by model-checking, the domain of testing and supervision, the region automaton is very used because it allows de-timing the specification [17].

Several verification tools are based on this correspondence between the timed model and its regions automaton.

This work focus on the transformation of the DATA* model into aggregate regions automata by the approach of graph transformation.

The graph transformation approach seems promising because it raises a lot of research and it gave birth to tools such as AToM3 [2] [9].

This study is based on an aggregation procedure of the regions automata to reduce the combinatorial explosion [16] of the regions.

AToM3 tool is used for the implementation of this application as well as the Python language [12]. A grammar is proposed to transform a graph of the specification DATA* into a graph of aggregate region automaton corresponding.

Each graph is a model which is represented by a meta-model as class diagrams.
The paper is organized as follows, after presenting the model of Durational action timed automata (DATA*), the classical regions automaton and the aggregation procedure of this latter is presented in sections (2), (3) and (4), section (5) deals with the transformation of models particularly the transformation of graphs as well as the tool AToM3 and Python language. In section (6) transformation rules are discussed, an example is dealt in section (7), finally we conclude with a discussion on the scope of this transformation and prospects for the use of this result.

2. DATA* MODEL

The DATA* model (Durational Action Timed Automata) [4] is a temporal model defined by a timed transitions system over an alphabet representing actions to be executed. This model takes into account in the specification, the duration of actions based on an intuitive idea: temporal and structural non-atomicity of actions.

This model seems interesting and funneling more and more research because it coated models of timed automata by maximality semantic [14]. The DATA* model, as the temporized models takes in charge the notions of urgency and deadlines as temporal constraints of the system.

Illustrate this model with examples:

\[
\begin{align*}
S_1 & \xrightarrow{a, x:=0} S_2 & (x \geq 2) \quad & b, y:=0 & (y \geq 10) \\
S_2 & \xrightarrow{x \geq 2} S_0
\end{align*}
\]

Figure 1: DATA* (a)

The durations associated to the actions are represented by constraints on the transitions and in the states targets of each of them. In this sense, any enabled transition represents the beginning of the execution of the action. On the target state of transition, a timed expression manifest that the action is possibly under execution. From operational point of view, for each action is associated a clock which is resets to 0 at the start of the action, this clock will be used in the construction of the temporal constraints as guards of the transitions.

The figure1 presents a system of two consecutives actions a and b, the x clock is associated to the action a, on the locality s1, the temporal expression \(\{x \geq 2\}\) represents the duration of the action a. The end of the execution of an action is deduced implicitly in the case of an action that it is causally dependent. The b action depends of a, so the transition is guarded by the relative constraint with duration of a.

Consider another example, the system P which consists of two concurrent processes P1 and P2 synchronizing on an action d. The process P1 executes the action a followed by d, while P2 executes b then d, and suppose that actions a, b and d have respectively the durations 10, 12 and 4.

The behaviour of P is given by the DATA* of the figure2 [4]. From the state s0, actions a and b can comply in parallel, and each one can finish only if its clock reaches a value equal to its duration, from where duration condition set \(\{x \geq 10, y \geq 12\}\).

The guard of the action d becomes \(x \geq 10 \land y \geq 12\). Once this latter is satisfied, d can start at any time in the enabling open interval \(x \in [10, +\infty[, y \in [12, +\infty[, \text{ the so called enabling domain.}\)

Other concepts of real time systems as the deadlines and the urgency [5] are considered.

\[
\begin{align*}
S_0 & \xrightarrow{a} S_1 & (x \geq 10) \quad & b, y:=0 & (y \geq 12) \\
S_0 & \xrightarrow{x \geq 2} S_2
\end{align*}
\]

Figure 2: DATA* (b)

2.1. FORMALIZATION

\textbf{Definition 2.1:} A DATA* A is a tuple \((\text{ACT}, S, s_0, H, T_D, L_S)\) where \text{ACT} is a finite set of symbols for actions, \(S\) is a finite set of states, \(s_0 \in S\) is the initial state, \(H\) is a finite set of clocks and \(T_D\) is the set of transitions. A transition \((s, G, D, a, x, s')\) represents switch from state \(s\) to state \(s'\) by starting execution of action \(a\) and resetting clock \(x\). \(G\) is the corresponding guard which must be satisfied to fire this transition. \(D\) is the corresponding deadline which requires, at the moment of its satisfaction, that action \(a\) must occur.

\(L_S : S \rightarrow 2^{L_S}\) is a function which corresponds to each state \(s\). \(F\) is the set of actions conditions potentially executed in \(s\).

\textbf{Definition 2.2:} The semantics of a DATA* A is defined by associating to it an infinite transitions system \(S_A\) over \text{ACT} \cup \text{T}. 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 over \(H\). 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 configurations of $S_A$ are possible and correspond respectively to time passing (rule RA1* and RA2*) thus the launching of a transition from $A$ (rule RD*).

Where $\eta$ is the smallest real quantity of time during which no action occurs.

3. REGION AUTOMATA

A region automaton is a finite representation of timed automata. Also it is well known that in the domains of verification by model checking or testing, the region automaton is used because it de-timed the specification.

We resume in this section the classical definition of regions automata [1] [7].

3.1 CLOCK REGIONS

A region is a set of valuations of a finite set of clocks, such as from two valuations of the same region, the same transitions are firable.

Definition 3.1: Let $A$ be a DATA*, the set of standard regions of $A$ is the set of relation equivalence classes defined over the clocks valuations as follows:

$$\varpi \equiv \varpi', \text{if } 8(\xi; \psi) \leq H$$

1. ($b(\varpi(\xi)) \leq b(\varpi(\psi))$ or ($\varpi(\xi) > M_\xi$) et ($\varpi(\psi) > M_\psi$))
2. ($\varpi(\xi) \leq M_\xi$ et $\varpi(\psi) \leq M_\psi$) )

\[
\left( \frac{\varpi(\xi)}{\varpi(\psi)} \leq \frac{\varpi(\psi)}{\varpi(\psi)} \right) \] \[
\frac{\varpi(\xi)}{\varpi(\psi)} \leq \frac{\varpi(\psi)}{\varpi(\psi)}
\]

$M_\xi$ are the maximum constants appearing in the constraints of clock $x$, and for any real $\tau$, $b(\tau)$ denotes the integral part of $\tau$ and $\text{frac}(\tau)$ denotes the fractional part of $\tau$. We note $\Delta$ the set of regions.

For example if we consider a set $H$ consisting of two clocks $x$, $y$ and $M_x = 3, M_y = 1$. So we have 38 regions: (Figure 3)

3.2 REGION AUTOMATA

Definition 3.2: Let $A$ be a timed automaton defined by a timed transition system, the region automaton $R_A$ corresponding to $A$ is a finite automaton defined as follows:

All localities of $R_A$ are of the form $(s, r)$ where $s$ is a state and $r$ is a clock region. The initial locality is $(s_0, r_0)$.

The set of transitions $T_R$ is,

$T_R = \left\{ t'(s', r') \rightarrow t(s, r) \leq \frac{\varpi(\xi)}{\varpi(\psi)} \right\}
\text{such as } r \subseteq g \text{ and } r' = r[t \leftarrow 0]\]

We present an example of the region automaton associated to a DATA* in figure 4.
4. AGGREGATE REGION AUTOMATA

An aggregation operation on the localities of a region automaton has been defined in [16], for timed models based on a BI-relation between the localities of the region automaton. This operation consists to substitute the grouped localities by a locality having the same name of state and a region that includes several regions successors. Localities are regrouped iff they have the same transitions “forward mirror” and “backward mirror”. The aggregation operation revealed aspects of symmetry for clocks regions when creating the region automaton associated to DATA*. In fact, due to the causal dependence of actions witch results from the consideration of the actions durations, the guards of transitions have a particular form relative to the beginning of each execution of an action; with these two characteristics we can deduce the form of all regions and its temporal successor verifying the guard and the reset of a given transition.

The grouping of localities (s, r_i) consists on the creation of a new locality (s, R) where R is the summation over regions r_i of localities to regroup.

**Note:** The summation of clocks regions is an operation defined on the set of regions as follows:

\[ R = \sum (r_1, r_2, ..., r_k) = \bigcup \bigcup \bigcup_{i=1}^{k} r_i \]

It was proved in [16] for the model of DATA*, from the specification we can generate an aggregate region automaton by transformation. The rules of transformation are based on the translation of the states and reset clocks to the destination state of transition.

For any transition \( t = (s, g, d, a, x, s') \) \( \in T_D \) of DATA* (A), correspond a transition \( t' = ((s, r_i), a, (s', g \land d \land x = 0)) \) \( \in T_R \) from the initial locality \( (s_0, r_0) \) such as \( s_0 \) is the initial state of A and \( r_0 = (\forall x \in H, x = 0) \).

Example, figure 5 presents the aggregate region automaton associated to DATA* A of figure 4.

5. 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 operation takes the source model in input then executes the rules of transformation and generates the target model in output.

Graph Grammars [13] are used for model transformation [3] [8] [10]. 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 sub graph in the host graph, then the rule can be applied and the matching sub graph 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. ATOM3 [2] [9] is a graph transformation tool among others. In this paper we use it.

**Example of Grammar Rule in ATOM3**

6. THE APPROACH

This approach based on two meta-models, one for the model of durational action timed automata and the second for the aggregate regions automata thus a grammar for generating an aggregate region automaton from a DATA*.
6.1 DATA* META-MODEL

The first meta-model proposed is a class diagram composed of the following classes (figure 7):

- **The class DATAct**: this class represents the states of DATA*, each state has two attributes that are name (name), and conditions of duration (CD).

It is connected to the association TransitionD and to the class DATActInit by an inheritance link.

- **The association TransitionD**: Represents the transitions of DATA*, each transition is identified by an action, a clock, a guard and a deadline.

- **The class DATActInit**: This class represents the initial state of DATA*, it inherits the attributes of class DATAct.

![Figure 7: DATA* Meta-Model](image)

6.2 AGGREGATE REGION AUTOMATON META-MODEL

The second meta-model is a class diagram composed of the following classes (figure 8):

- **The Class ARstate**: this class represents the localities of the aggregate region automaton, each locality has two attributes that are name (name), and a clock region (RegionHorloge). It is connected to the association ARtransition and to the class ARStateInit by an inheritance link.

- **The Association ARtransition**: represents the transitions of the aggregate region automaton, each transition is identified by an action.

- **The Class ARStateInit**: this class represents the initial locality of the aggregate region automaton; it inherits the attributes of the class ARstate.

Each class has a unique graphical appearance.

![Figure 8: Aggregate Region Automaton Meta-Model](image)

6.3 MODELING TOOL OF DATA* AND THE AGGREGATE REGION AUTOMATON

The two meta-models as defined are represented in AToM3 (figure 7, figure 8) and will allow us to generate a tool for modeling systems in DATA* and the aggregate region automaton.

Figure 9 represents the graph of an automaton DATA* modeling a system and its aggregate region automaton after the application of transformation rules.

![Figure 9: Modeling Tool of DATA* and the Aggregate Region Automaton](image)

6.4 THE GRAPH GRAMMAR PROPOSED

The proposed grammar allows generating aggregate region automata; it is composed of seven rules fall into three categories:

In the first 3 rules, we apply the principle of constructing an aggregate region automaton detailed in section 4; these rules treat the case of parallelism, choice and the case of sequential actions:

- The 1st rule is used to generate the first region associated to the initial state of DATA * where all clocks are reset to zero (Figure 10).
7. EXAMPLE

To illustrate the proposed approach we use a simple example frequently encountered in the literature called the double click of a mouse, where the system displays a simple if no second click occurs in the interval [0,1] of time, otherwise, it displays a double.

We present the result of execution of the grammar defined above, which realize the correspondence between the DATA* of double click (Figure 14) and the aggregate region automaton (Figure 15.a). The result will be saved in a text file (Figure 15.b).

8. CONCLUSION

In this paper we propose a method for generating an aggregate region automaton from a DATA* by the graph transformation approach and using the environment AToM3.

The aim of this transformation is to provide a finite abstraction of a model DATA*. Initially, it is suggested an automated approach based on the model transformation techniques, using a graph grammar, to transform a DATA* into an aggregate region automaton then drawing from the concept of meta-modeling and the principle of constructing an aggregate region automaton.

Finally, we defined the format of a text file with well-defined syntax to retrieve the aggregate region automaton resulting.

REFERENCES:


