Baltic J. Modern Computing, Vol. 3 (2015), No. 4, 225-233

Equivalent States Search Algorithm for Models with Continuous Time

Dalius MAKACKAS, Regina MISEVIČIENĖ

Kaunas University of Technology, K.Donelaicio 73, LT-44029 Kaunas, Lithuania

dalius.makackas@ktu.lt, regina.miseviciene@ktu.lt

Abstract. Verifying correctness of real time systems, reachable states analysis method is amply developed and used. However, this method described in scientific literature cannot avoid the endless increase of reachable state space. This paper presents an equivalent nodes search algorithm enabling to reduce the number of nodes in the reachable states graph. The application of this algorithm allows transforming the graph of the infinite reachable states into the graph with the finite node numbers. An example illustrates the algorithm.

Keywords: real time systems, continuous time models, verification, piece-linear aggregate formalism

1. Introduction

The systems whose functioning depends on the time constraints are called the real time systems (Logothetis, 2004; Kopetz, 2011). Strict time limitations and timely solutions of serious problems are the main factors describing the real time systems. The importance of these factors is described in the definition given in the paper (Silberschatz, 1998). Referring to the definition, the real time system properly functions only in case when it returns the correct result and performs the right actions during the defined time conditions. The definition shows that the real time systems have to process the information and give the answer during the determined time or otherwise it can cause the serious problem. Consequently, time performs an important role (Bonhomme, 2013; Furia, 2008; Logothetis, 2004; Pranevicius, 2008; Luckham, 2011).

Modeling the real time system functioning continuous time models are often used. The continuous time models use the real numbers domain for the time. They are the best for the modeling of the real time systems (Logothetis, 2004).

Various methods of the real time system analysis are used. One of them, such as a reachable states analysis method is widely developed and used for the real time system (Bouyer, 2010; Pranevicius, 2008). Any system can have thousands or even hundred thousands of reachable states. It is obvious that the system at any time moment is only in a single state. The target of the reachable state method is to generate and check all the system states that can be reached from the initial system state.

The main problem of this method is the exponential growth of the reachable state space (Knorreck, 2013). The state abstraction methods can help to solve this problem while trying to get the abstract system with the finite number of states.

The region graphs or simulation graphs are used in the real time models with continuous time as the abstraction method (Logothetis, 2004; Bouyer, 2010; Berard, 2010; Miao, 2006). These graphs are used for the mathematical formalization of time automate trying to transform the infinite states space into the finite one. However, practically the region graph model is not used as the number of states grows exponentially when the number of actions increases (Tripakis, 2005).

The aim of this paper is to present the algorithm enabling to decrease the number of nodes in the graph using the equivalent relationships, when the real time systems are described in piece-linear aggregate mathematical formalism. This formalism allows to fulfill correctness analysis based on the common formal specification and to make the simulation model of the analyzed system (Pranevičius, 2008). The correctness analysis is fulfilled using different validation and verification methods. One of them is the method of reachable states. The essence of this method is generation of the entire graph of system states. This graph is called the reachable states graph (RSG). Analysis of the graph enables the checking the specification correctness. The graph can help to analyze the characteristics of system.

Analysis of the composed RSG both for the discrete and continuous time models is rather difficult because of the infinite number of nodes in the graph. Thus, this paper presents an algorithm to decrease the number of nodes in the reachable states graph using the equivalent relationships. The algorithm enables to transform the issue graph into the smaller one. Using the reachable states graph received after the transformation, the same answers as in the initial graph can be achieved. This is especially important when RSG is made for the continuous time models.

The transformation algorithm of the reachable states graph for the continuous time models is given in the second part of the paper while the third part presents the small fragment showing how the algorithm performs.

2. Transformation algorithm of the reachable states graph

This work is a continuation of the previous article (Makackas, 2013). In the previous paper, the authors presented an algorithm how to make the reachable states graph in the models with the continuous time. When RSG is made according to this algorithm, then the received infinitive reachable states graph has the tree structure. This paper presents transformation of the infinitive tree to finite states graph.

Further, some definitions that will be used for transformation of the reachable state tree are presented.

Usually the system functioning is described by the states sequence:

$$\nu = s_1, s_2, s_3, \ldots. \tag{1}$$

An event that transfers the system is into another state is included trying to disclose the cause of the system state change. In this case, the system behavior is defined by the trajectory:

$$\sigma = s_0, e_0, s_1, e_1, s_2, e_2, s_3, e_3, \ldots \tag{2}$$

Other notion can be used:

$$s_0 \xrightarrow{e_0} s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} s_3 \xrightarrow{e_3} \ldots \tag{3}$$

The trajectory or the route is called finite if it is made of finite numbers of elements.

Rarely the real time system functions according to the strict event sequence because it is usually conditioned by various accidental changes. In this case the system functioning is described by the trajectory set Σ. Pointing out that the trajectories are analyzed beginning with the peculiar state, it can be written as:

$$\sigma(s_k) = s_k \xrightarrow{e_1} s_{k+1} \xrightarrow{e_2} \dots \quad (4)$$

The trajectory set from the state s_k is marked as $\Sigma(s_k)$ and it is called as the end of the trajectory $\sigma = s_0, e_0, s_1, e_1, s_2, e_2, s_3, e_3, \dots, e_k, s_k, \dots$ from the *k* transition in the sequence.

Definition 1. The trajectory is called periodical, if the two states s_i, s_j exist, where $\Sigma(s_i) = \Sigma(s_j), i \neq j$.

When all the trajectories are periodical or have the finite length, then Σ can be expressed by the finite graph, otherwise the graph has the tree structure.

When describing the system state change not only the event sequence should be known, but the time when the events take place as well. It is especially important for the real time systems. In this case, the system behavior is defined by the following trajectories that are called traces:

$$\rho = s_0, e_0(t_0), s_1, e_1(t_1), s_2, e_2(t_2), s_3, \dots \quad (5)$$

Other notion can be used:

$$s_0 \xrightarrow{e_0(t_0)} s_1 \xrightarrow{e_1(t_1)} s_2 \xrightarrow{e_2(t_2)} s_3 \xrightarrow{e_3(t_3)} \dots \quad (6)$$

Analyzing the trace $\rho = s_0, e_0(t_0), s_1, e_1(t_1), s_2, e_2(t_2), \dots$ by T time units from the modeling start time moment t_0, the events time will be also recalculated from the point of view of the time moment $t_0 + T$. The newly received trace is shown as follows:

$$\rho[T] = s_k, e_k(t_k - T), s_{k+1}, e_{k+1}(t_{k+1} - T), \dots, \text{ where } t_{k-1} < T < t_k. \quad (7)$$

Definition 2. The trace $\rho = s_0, \dots s_i, e_i(t_i), s_{i+1}, e_{i+1}(t_{i+1}), s_{i+2}, \dots s_j, e_j(t_j), s_{j+1}, e_{j+1}(t_{j+1}), s_{j+2}, \dots$ is called periodical, if two states s_i and s_j exist, where $\rho[t_i] = \rho\lfloor t_j \rfloor, i \neq j$.

Two traces $\rho^1 = s_0^1, e_0^1(t_0^1), s_1^1, e_1^1(t_1^1), \dots$ and $\rho^2 = s_0^2, e_0^2(t_0^2), s_1^2, e_1^2(t_1^2), \dots$ are considered equivalent when they matched the same trajectory, where $t_j^1 < t_{j+1}^2$ and $t_j^2 < t_{j+1}^1, \forall j > 0$.

The equivalent class generated by the traces is called behavior and is defined as follows:

$$\pi = s_0, e_0[I_0], s_1, e_1[I_1], s_2, e_2[I_2], \dots \quad (8)$$

Other notation is used:

$$s_0 \xrightarrow{e_0[I_0]} s_1 \xrightarrow{e_1[I_1]} s_2 \xrightarrow{e_2[I_2]} s_3 \xrightarrow{e_3[I_3]} \dots, \quad (9)$$

In the formula (9) I_j notes the time interval of the event e_j occurrence.

Two behaviors $\pi_1 = s_0^1, e_0^1[I_0^1], s_1^1, e_1^1[I_1^1], s_2^1, \ldots$ and $\pi_2 = s_0^2, e_0^2[I_0^2], s_1^2, e_1^2[I_1^2], s_2^2, \ldots$ are called equal when $s_j^1 = s_j^2$ (e_j^1 matches e_j^2), and their occurrence intervals are equal $I_j^1 = I_j^2$ for all j.

All the possible system behaviors marked as Π, and $\Pi(s_k)$ – define the behavior system from the state s_k or is called the end of the behavior $\pi = s_0, e_0[I_0], s_1, e_1[I_1], s_2, e_2[I_2] \ldots, e_k[I_k], s_k, \ldots$ from the k transition in the sequence.

Definition 3. The behavior π is called periodical when its traces are periodical.

The system detailed functioning can be shown by the graph when all the behaviors in the set Π are periodical or finite.

Having the behavior $\pi = s_0, e_0[I_0], s_1, e_1[I_1], s_2, e_2[I_2], \ldots$ and moving the modeling start to the time moment T, the system behavior will be described as follows:

$$\pi[T] = s_k, e_k[I_k^*], s_{k+1}, e_{k+1}[I_{k+1}^*], s_{k+2}, e_{k+2}[I_{k+2}^*], \ldots. \quad (10)$$

In the formula (10) $I_j^* = \left\{\max\{0, a-T\} \middle| a \in I_j\right\}$, $k = \max_j\left\{j+1 \middle| t_j < T, \forall t_j \in I_j\right\}$.

Definition 4. Two states s_i and s_j, are called equivalent in the behavior $\pi = s_0, e_0[I_0], s_1, e_1[I_1], \ldots, s_i, e_i[I_i], s_{i+1}, e_{i+1}[I_{i+1}], s_{i+2}, \ldots s_j, e_j[I_j], s_{j+1}, e_{j+1}[I_{j+1}], s_{j+2}, \ldots,$ when $\forall T_1 \in I_i, \exists T_2 \in I_j : \pi[T_1] = \pi[T_2]$.

Assume the analysis of behavior $\pi = s_0, e_0[I_0], s_1, e_1[I_1], \ldots, s_i, e_i[I_i], s_{i+1}, e_{i+1}[I_{i+1}], s_{i+2}, \ldots s_j, e_j[I_j], s_{j+1}, e_{j+1}[I_{j+1}], s_{j+2}, \ldots,$ that corresponds to any tree branch.

Seeking to check whether the states s_i and s_j are equivalent, there is necessary to check if the state discrete elements match. Fig. 1 presents the search algorithm of the equivalent states. Besides, there is necessary to check whether the continuous components describe the same further functioning of the system. As in the composing of RSG the absolute time is used then the values of the continuous components vary in those nodes. Trying to persuade that the continuous elements also match one should check whether this difference is "constant".

Definition 5. Assuming that in both states s_i and s_j are active the same operations O_1, O_2, …, O_n. The interval condition $\alpha_k^i < w(e_k, t) < \beta_k^i, \forall k = 1,2,\cdots,n$ in the state s_i satisfies the condition $\alpha_k^j < w(e_k, t) < \beta_k^j$, $\forall k = 1,2,\cdots,n$., in the state s_j only if the conditions $\left\{a - \min_l \alpha_l^i \middle| \alpha_k^i < a < \beta_k^i\right\} = \left\{a - \min_l \alpha_l^j \middle| \alpha_k^j < a < \beta_k^j\right\}$ are valid. Then the continuous components match.

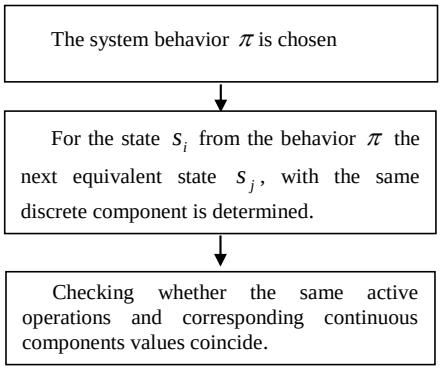

Fig. 1. The search algorithm of the equivalent states

3. The example of equivalent state search

The algorithm of the equivalent state search is illustrated by the small example. The system of two service equipment described in the previous paper (Makackas, 2013) is discussed. The mentioned paper depicts the algorithm estimating the time intervals where the events occurring in the system are defined. In this case, the tree node of the reachable states describes one system state that is made of three components:

- Discrete state component $\nu(t)$;
- Continuous state component $z_\nu(t)$;
- The time moment limitation sets R^+. This set elements are inequalities of the form $t_j - t_i < \alpha$, where α is the constant and t_j, t_i are time moments when the events occurred.

The arch of the graph connecting two nodes defines:

- When the transition can occur;
- Which event generates this transition.

The reachable states tree is composed using the algorithm of the reachable states graph described in the previous paper (Makackas, 2013). The composed tree of the reachable states has infinite number of nodes. It can be noticed that the same discrete and continuous node components start to repeat in the same branch, because they are similar from the point of view of the measurable period.

To illustrate the algorithm of equivalent states search presented in this paper the tree fragment made of 10 nodes was used (Fig. 2). Black arrows indicate that the tree goes into an infinite number of states.

The parameters of tree nodes from the previous article are these:

$1:(0;(t_0+4,t_0+6),\varnothing,\varnothing;R_0)$, where $R_0=\varnothing$

$2:(0;(t_1+4,t_1+6),(t_1+3,t_1+5),\varnothing;R_{11})$, where $R_{11}=R_0\cup\{t_0+4<t_1<t_0+6\}$

$3:(0;(t_1+4,t_1+6),\varnothing,\varnothing;R_{21})$, where $R_{21}=R_{11}\cup\{t_1+3<t_2<t_1+4\}$

$4:(0;(t_2,t_1+6),\varnothing,\varnothing;R_{22})$, where $R_{22}=R_{11}\cup\{t_1+4<t_2<t_1+5\}$

$5:(0;(t_2+4,t_2+6),(t_2,t_1+5),(t_2+2,t_2+4);R_{23})$,

where $R_{23} = R_{11} \cup \{t_1 + 4 < t_2 < t_1 + 5\}$

$6: (0; (t_3 + 4, t_3 + 6), (t_3 + 3, t_3 + 5), \varnothing; R_{31})$, where $R_{31} = R_{21} \cup \{t_1 + 4 < t_3 < t_1 + 6\}$

$7: (0; (t_3 + 4, t_3 + 6), (t_3 + 3, t_3 + 5), \varnothing; R_{41})$, where $R_{41} = R_{22} \cup \{t_2 < t_3 < t_1 + 6\}$

$8: (0; (t_2 + 4, t_2 + 6), \varnothing, (t_2 + 2, t_2 + 4); R_{51})$, where $R_{51} = R_{23} \cup \{t_2 < t_3 < t_1 + 5\}$

$9: (0; (t_3 + 4, t_3 + 6), \varnothing, \varnothing; R_{61})$, where $R_{61} = R_{31} \cup \{t_3 + 3 < t_4 < t_3 + 4\}$

$10: (0; (t_4, t_3 + 6), \varnothing, \varnothing; R_{62})$, where $R_{62} = R_{31} \cup \{t_3 + 4 < t_4 < t_3 + 5\}$

$l_{11} = (e_1, t_1 \in (t_0 + 4, t_0 + 6))$

$l_{21} = (e_2, t_2 \in (t_1 + 3, t_1 + 4))$

$l_{22} = (e_2, t_2 \in (t_1 + 4, t_1 + 5))$

$l_{23} = (e_1, t_2 \in (t_1 + 4, t_1 + 5))$

$l_{31} = (e_1, t_3 \in (t_1 + 4, t_1 + 6))$

$l_{41} = (e_1, t_3 \in (t_2, t_1 + 6))$

$l_{51} = (e_2, t_3 \in (t_2, t_1 + 6))$

$l_{61} = (e_2, t_4 \in (t_3 + 3, t_3 + 4))$

$l_{62} = (e_2, t_4 \in (t_3 + 4, t_3 + 5))$

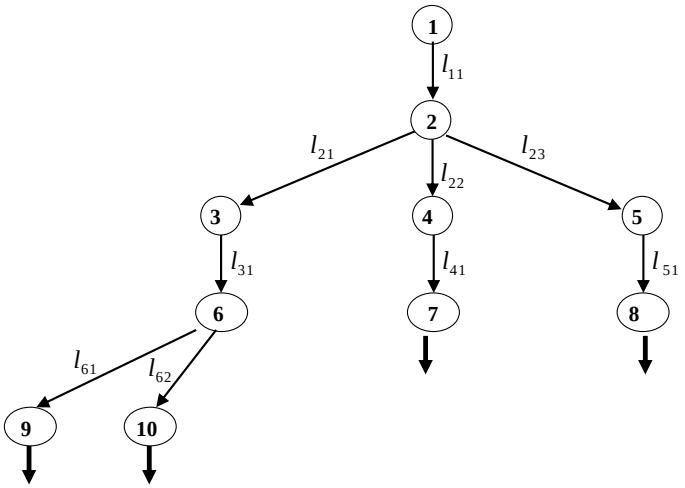

Fig. 2. Reachable state tree fragment

While studying RSG nodes it can be noticed that some nodes structure is identical. It differs only by time moments during the given event number. For example, discussing the structure of the nodes 3 and 9 it can be noticed that these nodes are equivalent. It enables to state that such nodes that repeat after some measurable time exist. Thus the tree of the reachable states can be transformed into the graph where the nodes are connected and the system functioning is shown up to repeating nodes saying that the system functioning is analogous to the equivalent nodes.

The equivalent states are searched in the reachable state tree in accordance with the described transformation algorithm of reachable state graph. It should be proved that the nodes 3 and 9 are equivalent:

$3:(0;(t_1+4,t_1+6),\varnothing,\varnothing;R_{21})$,

where $R_{21}=\{t_0+4<t_1<t_0+6\}\cup\ \{t_1+3<t_2<t_1+4\}$

$9:(0;(t_3+4,t_3+6),\varnothing,\varnothing;R_{61})$,

where $R_{61}=\{t_0+4<t_1<t_0+6\}\cup\{t_1+3<t_2<t_1+4\}\cup$
$\{t_1+4<t_3<t_1+6\}\cup\{t_3+3<t_4<t_3+4\}$.

The discrete components of these states are equal. It should be proved that the condition $\left\{a-\min_l\alpha_l^i\middle|\alpha_k^i<a<\beta_k^i\right\}=\left\{a-\min_l\alpha_l^j\middle|\alpha_k^j<a<\beta_k^j\right\}$ is valid for the continuous state components. In this case in the third node it is $\min\alpha_l^i\Rightarrow\min(t_1+4)=t_1+4$ and in the ninth node it is $\min\alpha_l^i\Rightarrow\min(t_3+4)=t_3+4$.

Examining the first continuous component of the third state we get $\{t_1+4,t_1+6\}=\{0;2\}$. Examining the first continuous component of the ninth state applying the condition we get $\{t_3+4,t_3+6\}=\{0;2\}$. It follows that the continuous components satisfy the condition, thus these states are equivalent.

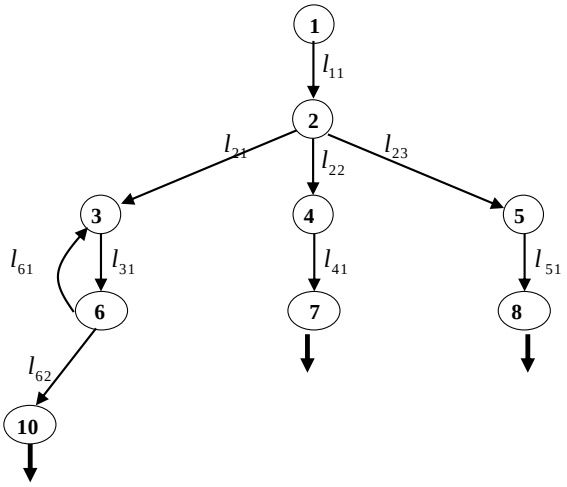

Fig. 3. Transformed graph

The transformed graph shown in Fig. 3 connects the discussed nodes.

All the rest tree nodes are studied analogically. The further calculations are fulfilled by the analogue thus, they are not presented. Finding all the equivalent states the reachable state tree was transformed into the reachable state graph.

4. Conclusions

Verifying correctness of real time systems, a reachable states graph analysis method is widely developed. Because of the infinite number of the nodes in the reachable states graph is impossible to analyze the graph.

This paper presented an algorithm that enables to minimize the nodes number of the reachable state graph using the equivalent relationships. Using the relations, the reachable states graph can be transformed into the graph with the finite nodes number.

It is especially important for continuous time models when transforming the graph into the smaller one and solving the same problems as in the original graph.

References

Berard B.; Bidoit M.; Finkel A.; Laroussinie F.; Petit A.; Petrucci L.; Schnoebelen P. (2001). *Systems and Software Verification: Model-Checking Techniques and Tools*, Springer -Verlag

Bonhomme, P. (2013). Scheduling and control of real-time systems based on a token player approach, *Journal of DiscreteEvent Dynamic Systems*, 23(2), 197-209

Bouyer, P.; Laroussinie, F. (2010). Model Checking Timed Automata, Merz S.; Navet N. (Eds), *Modeling and Verification of Real-Time Systems: Formalisms and Software Tools*, ISTE, London

Furia, C. A., Pradella, M., and Rossi, M. (2008). Automated verification of dense-time MTL specifications via discrete-time approximation. *In Proceedings of the 15th International Symposium on Formal Methods (FM'08)*, 132-147

Knorreck, D., Apvrille, L., Pacalet, R. (2013). Formal system-level design space exploration. *Concurrency and Computation: Practice and Experience*, 25, 250-264

Kopetz, H. (2011). *Real-time systems: design principles for distributed embedded applications*. Springer Science+ Business Media

Logothetis, G. (2004). *Specificication, Modelling, Verification and Runtime Analysis of Real Time Systems*, IOS Press

Luckham, David C. (2011). *Event Processing for Business: Organizing the Real-Time Enterprise*. Hoboken, New Jersey: John Wiley & Sons, Inc.

Makackas, D.; Miseviciene R.; Pranevicius H. (2013). Behavior Analysis of Real-Time Systems Using PLA Method, *Proceeding of Information and Software Technologies*, 392-402

Miao, H.; Xu, Q. (2006). *Constructing the Reaching Region Graph for Timed Automata with PVS*, IJCSNS, 6(7A), 175

Pranevicius, H. (2008). *Complex systems formalization and analysis*, Technologija, Kaunas

Silberschatz A., Galvin P.B. (1998). *Operating System Concepts*. Addison Weslwy Longman, Inc.

Tripakis, S., Yovine, S., Bouajjani, A. (2005) *Checking Timed Büchi Automata Emptiness Efficiently.* Formal Methods in System Design 26, 267–292

Authors' information

Dalius Makackas is a doctor of informatics sciences, an associated professor in the Department of Applied Informatics at Kaunas University of Technology. His main research interests include formal methods, real time systems and theory of algorithms.

Regina Misevičienė is a doctor of informatics sciences, an associated professor in Applied Informatics Department at Kaunas University of Technology. Her interests include creation of formal specifications using logic programming techniques, validation, and verification of protocols, business processes and knowledge bases.

Received August 5, 2015, accepted October 5, 2015