Formal Verification of a UML State Chart Diagram with Uppaal

Nianhua Yang[1,*], Xinshun Guo[1] and Wenjie Wang[2]

[1]*School of Business Information Management, Shanghai Institute of Foreign Trade, Shanghai 201620, China*
[2]*First Development Department, Shanghai Chuwa Software Co., LTD, Shanghai 200433, China*
yangnianhua@shift.edu.cn, gxs@shift.edu.cn, wangwje@gmail.com

**corresponding author*

Abstract

Semantics of UML state chart diagrams and timed automata are represented by timed transition systems. Based on the semantic equivalence, a method for transforming a state chart diagram into timed automata is proposed for the purpose of formally verifying requirements specification described by simplified CTL (Computation Tree Logic, CTL). The timed automata and simplified CTL is used as inputs of Uppaal for model checking.

Keywords: *UML, state chart diagram, verification, timed automaton, Uppaal*

1. Introduction

UML has become the de facto standard software modeling language. It includes use case diagrams, class diagrams, activity diagrams, state chart diagrams, et al. A state chart diagram specifies the reactive behavior of a class or the corresponding object. So it is widely used during software design phases.

Finding errors and conflicts in the early software development stage can significantly reduce entire development cost. In order to verify the functional correctness and analyze nonfunctional properties of a system in the early development phase, formal verification and analysis are necessary. Lacking of formal semantics, UML models can not be verified and analyzed formally.

Model checking [1] is an efficient verification method. It can be executed automatically and provide reversed error paths. A lot of model checking tools, such as Uppaal [2], have appeared. Uppaal has an easy-to-use graphical user interface and also applies for real-time system verification. Timed automata (TA) [3] are input models of Uppaal. Software requirements specifications are described with CTL (Computation Tree Logic, CTL) [1] in Uppaal.

This paper proposes an approach to transforming a UML state chart diagram and requirements specifications described with CTL into timed automata for formal verification with Uppaal.

2. Related Work

To meet the requirement of a more precise UML, several formalizing methods have been proposed for the behavioral of UML [4, 5, 6] and, in particular, the formalization of a state chart diagram [7, 8, 9, 10, 11]. Semantics of a state chart diagram is represented by model transition systems [12] in [10]. Lian et al. [8] propose a framework to transform UML state chart diagrams to Petri nets for dynamic semantics analysis. But Petri nets should also be transformed into input models of a model checker for verification.

Latella, et. al., [11] presents a methods for model checking a UML state chart diagram with SPIN [13]. Zhao et al. [9] propose mapping rules between a UML state chart diagram and Kripke structures [14] for model checking with NuSMV [15]. Uppaal provides a more user-friendly interface and applies to real-time system model checking. Furthermore, timed automata, input languages of Uppaal, hold similar semantics with UML state chart diagrams.

3. Preliminaries

3.1. Timed Automata

Let C is the set of clocks. j is clock constrains where $j := c \sim k \mid j_1 \wedge j_2$, $c \in C$, $k \in Q$ and $\sim \in \{<, \leq, ==, \geq, >\}$. Let $B(C)$ be the set of j. $u : C \to R_{\geq 0}$ assigns a non-negative real number to a clock variable. $\forall c \in C, u_0(c) = 0$.

A timed automaton [16] is a 6-tuple $TA = (L, l_0, C, A, I, E)$, where L is the finite set of locations, $l_0 \in L$ is the initial location, C is the set of clocks, A is the set of actions, $I : L \to B(C)$, $E \subseteq L \times A \times 2^C \times B(C) \times L$ is the set of states transitions, $< l, a, g, l' >$ represent an arc from l to l', $g \in B(C)$, l is the set of clocks which should be reset when the transition is finished.

Semantics of an automaton ($TA = (L, l_0, C, A, I, E)$) can be represented by a timed transition system [17] $TTS_A = < S, s_0, \to >$, where $S \subseteq L \times R_{\geq 0}^C$ is the set of states, $s_0 = (l_0, u_0)$ is the initial state, $\to \subseteq S \times \{R_{\geq 0} \cup A\} \times S$ consists of the following two transition relations:

1) $(l, u) \xrightarrow{d} (l, u + d)$, if $\forall d' : 0 \leq d' \leq d \Rightarrow u + d' \in I(l)$, where $d \in R_{\geq 0}$;

2) $(l, u) \xrightarrow{a} (l', u')$, if $\exists e = (l, a, g, l') \in E$, $u \in g$, $u' = u[l := 0]$ and $u' \in I(l')$, where $u[l := 0]$ put zero to each variable $c \in l$ in u.

3.2. Semantics of a State Chart Diagram

A state chart diagram is an 8-tuple $SD = < P, p_0, T, D, L, E, G, A >$, where P is the set of states with the initial state p_0, $T \subseteq P \times P$ is the set of transition arcs, E is the set of events, G is the set of guards, A is the set of actions, $L : T \to E \times G \times A$, $D : E \to \{[d_l, d_u] \mid d_l, d_u \in R^{\geq 0} \wedge d_l \leq d_u\}$ represents time interval domain of each event.

Semantics of a state chart diagram can be represented by a timed transition system $TTS_{SD} = (S, s_0, \rightarrow)$, where $S \subseteq P \times R_{s_0}^{C}$ is the set of states, $s_0 = (p_0, u_0)$ is the initial state, $\rightarrow \subseteq S \times \{R_{s_0} \cup E\} \times S$ consists of the following two transition relations:

1) $(p,u) \xrightarrow{d} (p,u+d)$, if $\forall d' : 0 \le d' \le d_l$;

2) $(p,u) \xrightarrow{e} (p',u')$, if $\exists e \in E$ and g is satisfied.

3.3. Model Checking

UPPAAL is designed to check whether a subset of CTL formulas are satisfied in the functional and structure models described by timed automata. Representation of a specification for a real-time system is represented by composition of a CTL formula and an observer automaton.

Simplified CTL is used in Uppaal. A simplified formula consists of path operators (A-all, E-exist), time operators ([]-all the future, <>-some time in the future) and predicate formulas. The formulas can be divided into five types [18].

4. Modeling a State Chart Diagram with Timed Automata

A transition in a state chart is driven by an external event or an event triggered by the execution of another transition (named action event) when the guard condition is satisfied. Execution of a transition will result in triggering some events. Key factors for transforming a state chart diagram into timed automata is how to model an external event, how to model a transition and how to model an event triggered by the execution of a transition.

A global urgent channel should be defined before defining any timed automata. The channel is named "*go*" in this paper. This channel ensure that the timed automaton commit a state transition after receiving the synchronous signal when other conditions are satisfied.

4.1. Define a Global Synchronous Signal Generator

Figure 1 defines a global synchronous signal generator. It generates synchronous signals continuously to ensure the enabled receiver commit a transition.

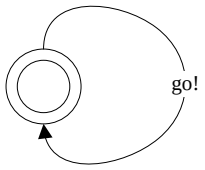

Figure 1. A Global Synchronous Signal Generator

4.2. Modeling an External Event

A transition in a state chart diagram may triggered by an external event when it is enabled. An external event of the same type may appear after some time interval. Figure 2 is a timed automaton for an external event in Uppaal. It shows that the event "*ee*" will be sent when the clock is between *dl* and *du* time units. The clock variable *c* is reset to zero after an event is triggered.

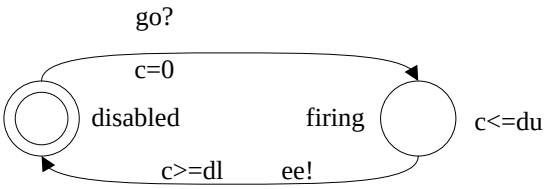

Figure 2. An External Event Model in Uppaal

4.3. Modeling a Transition and Action Events

An enabled transition in a state chart diagram will be executed once the synchronous event "*e*" appears. On the execution, an action will be taken and some events, such as "*ae*", will be generated. For some reasons, such as communication delay, the following state will receive updated signal after some period between "*dl*" and "*du*" time units. In Figure 3, the formula "*g=true*" represents that the enabled condition is satisfied; "*e?*" represents a receiving event; the arc marked with "*ae!*" represents creating an action event.

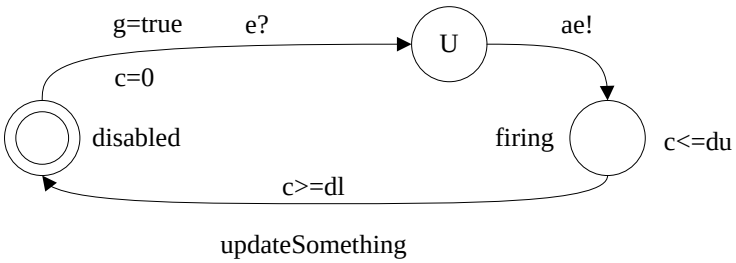

Figure 3. A Model for a Transition and its Created Action Events in Uppaal

5. Realization

In order to verify a state chart diagram with Uppaal, following steps should be taken:

1) Define a global urgent channel variable, named "go";

2) Define an automaton for generating a global synchronous signal according to the method in Section 4.1;

3) Define an automaton for each external event according to the method in Section 4.2;

4) Define an automaton for each transition in the state chart diagram according to the method in Section 4.3;

5) Define a CTL formula for each non-real-time requirement specification;

6) Define an observer automaton for each real-time requirement specification according to the tutorial on Uppaal [19].

6. Conclusions

This paper proposes a method to transform a state chart diagram into timed automata for requirement specification verification based on semantics equivalence. Events for triggering a transition are generated by timed automata. A UML state chart diagram can be verified formally. Dependability of the software model can be enhanced. We will develop a tool for automatic model transformation in the future.

Acknowledgements

The project is supported by Shanghai 085 Project for Municipal Universities and the Innovation Program of Shanghai Municipal Education Commission under grant No. 12ZS170.

References

[1] E. M. Clarke, E. A. Emerson and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications", ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 8, no. 2, **(1986)**, pp. 244-263.
[2] K. G. Larsen, P. Pettersson and W. Yi, "Uppaal in a nutshell", International Journal on Software Tools for Technology Transfer (STTT), vol. 1, no. 1, (1997), pp. 134-152.
[3] R. Alur and D. L. Dill, "A theory of timed automata", Theoretical Computer Science, vol. 126, no. 2, **(1994)**, pp. 183-235.
[4] P. S. Kaliappan and H. König, "On the formalization of UML activities for component-based protocol design specifications", Proceedings of the 38th international conference on Current Trends in Theory and Practice of Computer Science, Springer-Verlag, **(2012)**, Špindlerův Mlýn, Czech Republic, pp. 479-491.
[5] A. Queralt and E. Teniente, "Verification and validation of UML conceptual schemas with OCL constraints", ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 21, no. 2, **(2012)**, pp. 1-41.
[6] S. Distefano, A. Puliafito and M. Scarpa, "A representation method for performance specifications in UML domain", Computers in Human Behavior, vol. 27, no. 5, **(2011)**, pp. 1579-1592.
[7] E. Biermann, C. Ermel and G. Taentzer, "Precise semantics of EMF model transformations by graph transformation", Proceedings of the 11th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2008), Lecture Notes in Computer Science 5301, Springer Berlin / Heidelberg, **(2008)** September 28 - October 3, Toulouse, France, pp. 53-67.
[8] J. Lian, Z. Hu and S. Shatz, "Simulation-based analysis of UML statechart diagrams: Methods and case studies", Software Quality Journal, vol. 16, no. 1, **(2008)**, pp. 45-78.
[9] Y. Zhao, Z.-y. Yang and J. Xie, "Formal semantics of UML state diagram and automatic verification based on Kripke structure", Proceedings of the 22nd Canadian Conference on Electrical and Computer Engineering (CCECE '09), IEEE Computer Society, **(2009)** May 3-6, St. John's, NL, Canada, pp. 974-978.
[10] D. Varró, "A formal semantics of UML statecharts by model transition systems", Proceedings of the First International Conference on Graph Transformation, Lecture Notes in Computer Science 2505, Springer-Verlag, **(2002)** October 7-12, Barcelona, Spain, pp. 378-392.
[11] D. Latella, I. Majzik and M. Massink, "Automatic verification of a behavioural subset of UML statechart diagrams using the spin model-checker", Formal Aspects of Computing, vol. 11, no. 6, **(1999)**, pp. 637-664.
[12] D. Varró, G. Varró and A. Pataricza, "Designing the automatic transformation of visual languages", Science of Computer Programming, vol. 44, no. 2, **(2002)**, pp. 205-227.
[13] G. J. Holzmann, "The model checker SPIN", IEEE Transactions on Software Engineering, vol. 23, no. 5, **(1997)**, pp. 279-295.
[14] M. C. Browne, E. M. Clarke and O. Grümberg, "Characterizing kripke structures in temporal logic", Proceedings of the International Joint Conference on Theory and Practice of Software Development (TAPSOFT '87), Lecture Notes in Computer Science 249, Springer Berlin / Heidelberg, **(1987)** March 23–27, Pisa, Italy, pp. 256-270.
[15] A. Cimatti, E. Clarke, E. Giunchiglia, F. G. M. Pistore, M. Roveri, R. Sebastiani and A. Tacchella, "Nusmv 2: An opensource tool for symbolic model checking", Proceedings of the 14th International Conference on Computer Aided Verification (CAV 2002), Lecture Notes in Computer Science 2404, Springer Berlin / Heidelberg, **(2002)** July 27–31, Copenhagen, Denmark, pp. 241-268.

[16] R. Alur, "Timed automata", Proceedings of the 11th International Conference on Computer Aided Verification (CAV'99), Lecture Notes in Computer Science 1633, Springer, **(1999)** July 6–10, Trento, Italy, pp. 8-22.
[17] T. A. Henzinger, Z. Manna and A. Pnueli, "Timed transition systems, Proceedings of the Real-Time: Theory in Practice", REX Workshop, Lecture Notes in Computer Science 600, Springer, **(1992)** June 3–7, Mook, The Netherlands, pp. 226-251.
[18] J. Bengtsson and W. Yi, "Timed automata: Semantics, algorithms and tools", Lectures on concurrency and petri nets, Lecture notes in computer science 3098, J. Desel, W. Reisig and G. Rozenberg, eds., Springer Berlin / Heidelberg, vol. 3098, **(2004)**, pp. 87-124.
[19] G. Behrmann, A. David and K. G. Larsen, "A tutorial on Uppaal, International School on Formal Methods for the Design of Computer, Communication, and Software Systems", Lecture Notes in Computer Science 3185, Springer, **(2004)** September 13-18, Bertinora, Italy, pp. 200-236.