Development of Model Checker of Dynamic Linear Hybrid Automata

Ryo Yanase*, Tatsunori Sakai*, Makoto Sakai* and Satoshi Yamane†
*Graduate School of Natural Science and Technology
Kanazawa University, Kakuma-machi, Kanazawa 920–1192
Email: ryanase@csl.ec.t.kanazawa-u.ac.jp
†Institute of Science and Engineering
Kanazawa University, Kakuma-machi, Kanazawa 920–1192
Email: syamane@is.t.kanazawa-u.ac.jp

***Abstract*—Dynamically reconfigurable systems have attracted public attention from the point of view of miniaturization and saving power consumption for embedded systems in recent years. In this study, we propose *dynamic linear hybrid automata* as specification language of dynamically reconfigurable systems and the verification technique of reachability analysis. A dynamic linear hybrid automaton(DLHA) is a linear hybrid automaton extended with actions of creation and destruction. This paper presents the model checker and applies it to the model of an embedded system consisting CPU and DRP.**

***Keywords*-verification; model checking; hybrid automata; dynamically reconfigurable systems;**

I. Introduction

Embedded systems have been needed in recent years. The number of processors embedded into a system is increasing, and they cause troubles for miniaturization and conservation of electric power. Thus, dynamically reconfigurable processor(DRP) has been brought to public attention[1]. DRP executes a plural number of exclusive processes in the same board by dynamically changing the circuit configuration[2]. Generally, a DRP is used to accelerate computations of a CPU and loosely connected with the CPU. In a dynamically reconfigurable system, CPU behaves as a real-time system since a deadline is set in a task on the CPU. In addition, DRP behaves as a hybrid system with a dynamic change of operating frequency[3].

We model the embedded system, in which CPU and DRP cooperatively process a job. The system have been specified as a static system based on hybrid automata, but the state-space explosion problem has caused at the verification stage[4]. We present *dynamic linear hybrid automaton*(DLHA) and verification technique. Also, we have developed the model checker based on DLHA, and the results of several experiments are shown in this paper.

II. Specification

A. Dynamic Linear Hybrid Automaton

We define syntax and semantics of a dynamic linear hybrid automaton(DLHA) for specifying dynamically reconfigurable systems. A dynamic linear hybrid automaton is a linear hybrid automaton extended with particular actions

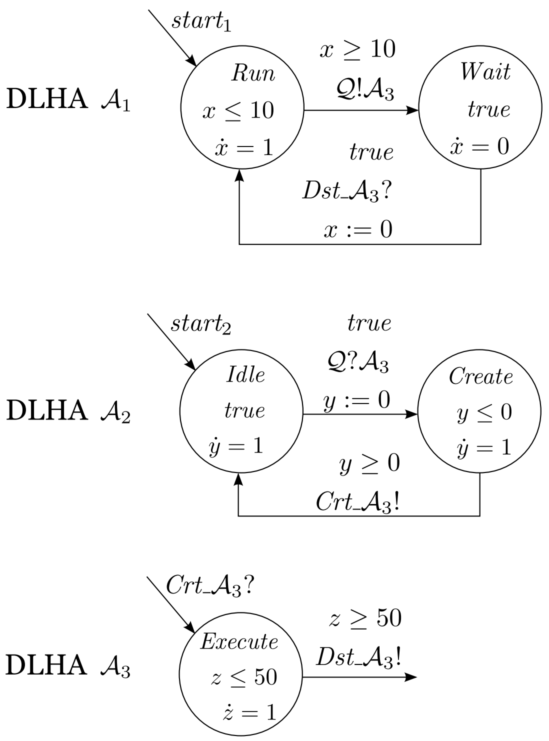

Figure 1. Dynamic linear hybrid automata

for creation and destruction of an automata[7], [8] and for operation of a queue. Fig. 1 shows examples of dynamic linear hybrid automata. Actions are described by CSP style.

$\mathcal{Q}!\mathcal{A}_3$ and $\mathcal{Q}?\mathcal{A}_3$ are actions for operating queue of the system. $\mathcal{A}_1$ sends the creation demand of $\mathcal{A}_3$ to the queue with an action $\mathcal{Q}!\mathcal{A}_3$ when it moves from location *Run* to location *Wait*. If head of the queue is the creation demand of $\mathcal{A}_3$, $\mathcal{A}_2$ dequeues the message and moves from location *Idle* to location *Create*.

*Crt_*$\mathcal{A}_3$! and *Crt_*$\mathcal{A}_3$? are special actions for creation of $\mathcal{A}_3$. $\mathcal{A}_2$ outputs *Crt_*$\mathcal{A}_3$! when it moves from location *Create* to location *Idle*, and $\mathcal{A}_3$ inputs *Crt_*$\mathcal{A}_3$? and is created by synchronizing with $\mathcal{A}_2$. Also, *Dst_*$\mathcal{A}_3$! and *Dst_*$\mathcal{A}_3$? are special actions for destruction of $\mathcal{A}_3$. When $\mathcal{A}_3$ outputs

$$\begin{tikzcd}
\sigma_0 :\ (\mathit{Run}, \mathit{Idle}),\ x = 0 \wedge y = 0,\ \text{``''} \arrow[d] \\
\sigma_1 :\ (\mathit{Wait}, \mathit{Idle}),\ x = 10 \wedge y = 10,\ \text{``}\mathcal{A}_3\text{''} \arrow[d] \\
\sigma_2 :\ (\mathit{Wait}, \mathit{Create}),\ x \geq 10 \wedge y = 0,\ \text{``''} \arrow[d] \\
\sigma_3 :\ (\mathit{Wait}, \mathit{Idle}, \mathit{Execute}),\ x \geq 10 \wedge y = 0 \wedge z = 0,\ \text{``''} \arrow[d] \\
\sigma_4 :\ (\mathit{Run}, \mathit{Idle}),\ x = 10 \wedge y = 50,\ \text{``''} \arrow[d] \\
\sigma_5 :\ \underbrace{(\mathit{Wait}, \mathit{Idle})}_{\text{Locations}},\ \underbrace{x = 10 \wedge y = 60}_{\text{Zone}},\ \underbrace{\text{``}\mathcal{A}_3\text{''}}_{\text{Queue}} \arrow[uuu, bend right=60]
\end{tikzcd}$$

Figure 2. Example of State-space: The state-space of DLHAs in Fig. 1

*Dst*_$\mathcal{A}_3$! and is destroyed, $\mathcal{A}_1$ inputs *Dst*_$\mathcal{A}_3$? and moves from *Wait* to *Run*.

Thus, we can specify the system as a dynamic system using DLHAs.

III. Verification

Verification of DLHAs is performed based on Alur's technique for symbolic verification of linear hybrid automata[7]. To compute reachable states, we add and delete variables by composing locations when a DLHA is created and destroyed. In addition, we represent state-transition systems by using *QDD(queue-content decision diagram)*[9].

For example, the result of searching the state-space of DLHAs in Figure 1 is shown in Figure 2. The initial state σ_0 is composed of DLHA $\mathcal{A}_1$ and DLHA $\mathcal{A}_2$ since DLHA $\mathcal{A}_3$ is not running. In the state σ_2, $\mathcal{A}_2$ creates $\mathcal{A}_3$ with the transition outgoing from location *Idle*, and the next state σ_3 is constructed by $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$. Also, $\mathcal{A}_3$ is destroyed with the transition outgoing from location *Execute* in the state σ_3, then σ_4 is constructed by $\mathcal{A}_1$ and $\mathcal{A}_2$. At that time, the zone of σ_4 is computed by removing a variable z from the zone of σ_3

IV. Practical Experiment

We have implemented a model checker of DLHA and a "QDD-package" in Java. Our experimental environment consists of dual processor machine (Xeon CPU 2.00 GHz) running Gentoo Linux 3.7.10 and Java version 7 update 10.

For example, consider a dynamically reconfigurable system consisting of a CPU and a DRP[4]. The CPU has two tasks, and they cooperate with two co-tasks on the DRP to process jobs. We have specified the system with DLHAs, and we have checked some properties.

Schedulability has been checked in 336.48 seconds with 214 MB RAM. In this case, the number of states was 1038 in total.

V. Conclusion and Future work

We have developed a model checker of dynamic linear hybrid automata(DLHA). In this paper, we have presented our tool and have shown an experimental result. The next step would be to compare our tool with other work in order to assesses its effectiveness.

References

[1] M. Motomura, T. Fujii, K. Furuta, K. Anjo, Y. Yabe, K. Togawa, J. Yamada, Y. Izawa and R.Sasaki. New Generation Microprocessor Architecture (2):Dynamically Reconfigurable Processor(DRP). *IPSJ Magazine*, Vol.46(11), pp.1259-1265, 2005.

[2] V. M. Tuan, Y. Hasegawa, N. Katsura and H. Amano. Performance Evaluation of Hardware Multi-process Execution on the Dynamically Reconfigurable Processor. *Technical Report of IEICE*, Vol.106, pp.25-30, 2006.

[3] H. Amano, Y. Adachi, S. Tsutsumi and K. Ishikawa. A Context Dependent Clock Control Mechanism for Dynamically Reconfigurable Processors. *Technical Report of IEICE*, Vol.104(589), pp.13-16, 2005.

[4] S. Minami, S. Takinai, S. Sekoguchi, Y. Nakai and S. Yamane. Modeling, Specification and Model Checking of Dynamically Reconfigurable Processors *Computer Software*, Vol.28(1), pp.190-216, 2011.

[5] Y. Hasegawa, H. Amano, S. Abe, S. Kurotaki and V. M. Tuan. Performance and Power Analysis of Time-multiplexed Execution on Dynamically Recofigurable Processor. *Technical Report of IEICE*, Vol.105(287), pp.31-36, 2005.

[6] V. M. Tuan and H. Amano. A Mapping Method for Multi-Process Execution on Dynamically Reconfigurable Processors. *IEICE Transactions on Information and Systems*, Vol.11, pp.2312-2322, 2008.

[7] R. Alur, C. Courcoubetis, T. A. Henzinger and P. Ho. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. *LNCS*, Vol.736, pp.209-229, 1992.

[8] P. C. Attie and N. A. Lynch. Dynamic Input/Output Automata: A Formal Model for Dynamic Systems. *LNCS*, Vol.2154, pp.137-151, 2001.

[9] B. Boigelot and P. Godefroid. Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs. *Formal Methods in System Design*, Vol.14, pp.237-255, 1999.