Abstraction refinement for non-zeno fairness verification of linear hybrid automata

Abstraction Refinement for Non-Zeno Fairness Verification of Linear Hybrid Automata

Ryo Yanase
Graduate School of Natural Science and Technology,
Kanazawa University, Kanazawa, Ishikawa 920–1192, Japan
Email: ryanase@csl.ec.t.kanazawa-u.ac.jp

I. Introduction

Linear hybrid automaton is a specification language for hybrid systems. For verification of hybrid systems, it is important to check fairness assumptions. For example, an embedded system keeps running forever when it starts to move by turning on the switch. Such a system has to be checked not only system safety but also fairness and non-Zenoness.

The state space explosion is a fundamental problem in model checking, since it is a method that performs an exhaustive search of states. To avoid state space explosion problem in model checking, CEGAR (CounterExample Guided Abstraction Refinement) is an effective technique. In this paper, we propose transition predicate abstraction and CEGAR verification algorithm for linear hybrid automata.

II. Related Work

Hybrid automaton is a specification language proposed by R. Alur, et al [1], and linear hybrid automaton is a subset of the language. R. Alur proposed predicate abstraction for reachability analysis and CEGAR verification algorithm [2], [3]. F. Wang specified non-Zeno fairness assumption by using extended temporal logic TECTL^f and proposed verification method for timed automaton [4]. R. Kurki-Suonio formalized a hybrid system as a discrete time system that has distributed clocks, and he verified non-Zeno fairness for the system [5]. B. Cook, A. Podelski and A. Rybalchenko extended predicate abstraction to *transition predicate abstraction* and proposed an algorithm for checking fair termination of discrete systems [6], [7].

In our work, we propose verification method by CEGAR using transition predicate abstraction for linear hybrid automaton. Our proposed CEGAR based on transition predicate is quite different from existing CEGARs based on state predicate.

III. Non-Zeno Fairness

Definition 1 (Non-Zeno fairness)**.** *Wang defined non-Zeno fairness by using event predicate and TECTL^f formula[4] as*

$$\exists^{[a_1,\ldots,a_m]}_{\langle b_1,\ldots,b_n\rangle}\Box^{\varepsilon}\varphi_1,$$

where $a_1,\ldots,a_m,b_1,\ldots,b_n,\varepsilon$ are either event predicates or TECTL^f formulas, and φ_1 is either null (not specified) or an event predicate. $[a_1,\ldots,a_m]$ denotes the strong-fairness and $\langle b_1,\ldots,b_n\rangle$ denotes the weak-fairness, and $\exists^{[a_1,\ldots,a_m]}_{\langle b_1,\ldots,b_n\rangle}$ means that there exists a path satisfying them.

- *The strong-fairness is a property that a transition satisfying an event predicate occurs at least once if the transition can be taken infinitely often.*
- *The weak-fairness is a property that a transition satisfying an event predicate occurs at least one if the transition can always occur.*

$\Box^{\varepsilon}\phi_1$ means that the path always satisfies ϕ_1 after a transition satisfying ε. If ε is not assigned, it is equal to a "globally" formula of typical CTL. In addition, the path must satisfy that the sum of time for each transitions diverges to infinity.

IV. Transition Predicate Abstraction for Linear Hybrid Automaton

We utilise Podelski's predicate abstraction method and extend it for linear hybrid automata.

A. Transition Predicate

Let a linear hybrid automaton be $H=(L,X,F,I,A,T,\theta)$, a transition predicate p is a binary relation over states, and it is described as an assertion over X and their primed variables.

1) Predicates of Transition Relations: A time transition in a location l of a linear hybrid automaton $H=(L,X,F,I,A,T,\theta)$ corresponds to the following set $\mathcal{P}_l^{\uparrow}$ of predicates:

$$\begin{aligned}\mathcal{P}_l^{\uparrow} &= \{at\_l, at'\_l\} \cup \mathrm{Preds}(I(l))\\ &\cup \mathrm{Preds}(\exists d\in\mathbb{R}_{\geq 0}.\bigwedge\{x_i' = x_i + F(l)(x_i)\cdot d \mid x_i\in X\})\\ &\cup \mathrm{Preds}(I(l)[X'/X]),\end{aligned} \tag{1}$$

where Preds() is a function that symbolically evaluates a formula and divides it into the set of atomic formulas with conjunctions (e. g. $\mathrm{Preds}(p_1\wedge p_2)=\{p_1,p_2\}$), and $[X'/X]$ denotes a formula computed by replacing $x_i\in X$ with the primed variable x_i'.

A discrete transition $\tau=(l,\phi,a,\lambda,l')\in T$ of H is also represented by the following set $\mathcal{P}_\tau$ of predicates:

$$\mathcal{P}_\tau = \{at\_l, at'\_l'\}\cup\mathrm{Preds}(I(l)\wedge\phi\wedge\lambda\wedge I(l')[X'/X]). \tag{2}$$

2) Composition of transition predicates: Given two sets $\mathcal{P}_1$ and $\mathcal{P}_2$ of transition predicates, the composition $\mathcal{P}_1 \circ \mathcal{P}_2$ is computed by the following rules:

- If $\bigwedge \mathcal{P}_1 = \mathit{false}$, $\mathcal{P}_1 \circ \mathcal{P}_2 = \{\mathit{false}\}$.
- If $\bigwedge \mathcal{P}_2 = \mathit{false}$, $\mathcal{P}_1 \circ \mathcal{P}_2 = \{\mathit{false}\}$.
- Let $\mathcal{P}_1 = \{at\_l_1, at'\_l'_1\} \cup \mathcal{P}'_1$ and $\mathcal{P}_2 = \{at\_l_2, at'\_l'_2\} \cup \mathcal{P}'_2$, If $l'_1 = l_2$,

$$\mathcal{P}_1 \circ \mathcal{P}_2 = \{at\_l_1, at'\_l'_2\} \cup \mathrm{Preds}((\exists X'.$$

$$(\rho_2[X''/X'])[X'/X] \wedge \rho_1)[X'/X'']),$$

where $\rho_1 = \bigwedge \mathcal{P}'_1$ and $\rho_2 = \bigwedge \mathcal{P}'_2$.
Otherwise, $\mathcal{P}_1 \circ \mathcal{P}_2 = \{\mathit{false}\}$.

Let $\mathcal{P}^{\uparrow}_{\tau}$ be a set of transition predicates when a discrete transition τ occurs after time transition from a location l, it can be computed as $\mathcal{P}^{\uparrow}_{l} \circ \mathcal{P}_{\tau}$, where $\mathcal{P}^{\uparrow}_{l}$ is the predication set of a time transition and $\mathcal{P}_{\tau}$ is one of a discrete transition.

Given a sequence $\pi = \tau_0\tau_1 \ldots \tau_k$ of discrete transitions, the predicate set $\mathcal{P}^{\uparrow}_{\pi}$ considering time transitions is computed as

$$\mathcal{P}^{\uparrow}_{\pi} = \mathcal{P}^{\uparrow}_{\tau_0} \circ \mathcal{P}^{\uparrow}_{\tau_1} \circ \cdots \circ \mathcal{P}^{\uparrow}_{\tau_k}.$$

3) Transition Abstraction for a Linear Hybrid Automaton: Given a linear hybrid automaton $H = (L, X, F, I, A, T, \theta)$ and a set $\mathcal{P}$ of transition predicates, the predicate abstraction for a predicate set $\mathcal{P}_{\tau}$ with $\mathcal{P}$ is defined as

$$\alpha_{\mathcal{P}}(\mathcal{P}_{\tau}) = \{p \in \mathcal{P} \mid \rho_{\tau} \subseteq p\},$$

where predicates at_l and at'_l for each location are implicitly contained by $\mathcal{P}$, and ρ_{τ} is an assertion of τ, that is, $\rho_{\tau} = \bigwedge \mathcal{P}^{\uparrow}_{\tau}$.

A predicate abstraction $\hat{\alpha}_{\mathcal{P}}(\pi)$ for a finite path $\pi = \delta_{d_0}\tau_0\delta_{d_0}\tau_1 \ldots \delta_{d_k}\tau_k$ is recursively computed:

$$\hat{\alpha}_{\mathcal{P}}(\pi) = \alpha_{\mathcal{P}}(\mathcal{P}^{\uparrow}_{\tau_0} \circ \hat{\alpha}_{\mathcal{P}}(\delta_{d_1}\tau_1 \ldots \delta_{d_k}\tau_k)$$
$$\hat{\alpha}_{\mathcal{P}}(\delta_k\tau_k) = \alpha_{\mathcal{P}}(\mathcal{P}^{\uparrow}_{\tau_k}).$$

For a sequence $\pi = \tau_0\tau_1 \ldots \tau_k$ of discrete transitions, the abstraction $\hat{\alpha}^{\uparrow}_{\mathcal{P}}(\pi)$ considering time transitions is computed as

$$\hat{\alpha}^{\uparrow}_{\mathcal{P}}(\pi) = \alpha_{\mathcal{P}}(\mathcal{P}^{\uparrow}_{\tau_0} \circ \hat{\alpha}^{\uparrow}_{\mathcal{P}}(\tau_1 \ldots \tau_k))$$
$$\hat{\alpha}^{\uparrow}_{\mathcal{P}}(\tau_k) = \alpha_{\mathcal{P}}(\mathcal{P}^{\uparrow}_{\tau_k}).$$

V. CEGAR for Non-Zeno Fairness

We propose the approach of non-Zeno fairness verification with CEGAR for a linear hybrid automaton based on Cook's method[7].

For proving the non-Zeno fairness $\exists^{[a_1,\ldots,a_m]}_{\langle b_1,\ldots,b_n\rangle}\Box^{\varepsilon}\phi_1$, we have to find the path satisfying the assumption. However, the search will not terminate if the number of paths is infinite. Avoiding this problem, we compress paths to check by using the transition abstraction method.

The algorithm of non-Zeno fairness verification with CEGAR is shown in Fig. 1.

1: $\mathcal{R} := \varnothing$ /* A set of ranking relations */
2: $\mathcal{P} := \varnothing$ /* A set of transition predicates */
3: **while** exists $\pi_2 = \tau_k \ldots \tau_l$ s.t. $\hat{\alpha}^{\uparrow}_{\mathcal{P}}(\pi_2) \not\subseteq R$ for any $R \in \mathcal{R}$ **do**
4: **if** exists $R \in \mathcal{R}$ s.t. $\mathcal{P}^{\uparrow}_{\pi_2} \subseteq R$ **then**
5: /* Refinement */
6: $\mathcal{P}_{path} := \bigcup_{i\in\{k,\ldots,l\}} \mathcal{P}^{\uparrow}_{\rho_{\tau_i}\ldots\tau_l}$
7: $\mathcal{P}_{loop} := R \cup \bigcup_{i\in\{k,\ldots,l\}} \mathcal{P}^{\uparrow}_{\tau_i\ldots\tau_l} \circ R$
8: $\mathcal{P} := \mathcal{P} \cup \mathcal{P}_{path} \cup \mathcal{P}_{loop}$ /* Update $\mathcal{P}$ */
9: **else**
10: **if** π_2 is well-founded by the ranking relation R **then**
11: $\mathcal{R} := \mathcal{R} \cup \{R\}$
12: **else**
13: **if** exists $\pi_1 = \tau_1 \ldots \tau_{k-1}$ s.t. $\hat{\alpha}_{\mathcal{P}}(\pi_1\pi_2) \neq \mathit{false}$ **then**
14: **if** $\mathcal{P}^{\uparrow}_{\pi_1\pi_2} \neq \mathit{false}$ **then**
15: **if** A cycle time of π_2 diverges,
16: $\forall i \in \{1,\ldots,m\}\exists j \in \{k,\ldots,l\}.$ $(\exists X.\mathcal{P}^{\uparrow}_{\pi_1\tau_k\ldots\tau_j})[X/X'] \wedge a_i \neq \mathit{false}$,
17: $\forall i \in \{1,\ldots,m\}.\forall j \in \{k,\ldots,l\}.$ $(\exists X.\mathcal{P}^{\uparrow}_{\pi_1\tau_k\ldots\tau_j})[X/X'] \wedge b_i \neq \mathit{false}$, and
18: $\forall i \in \{1,\ldots,l\}.$ $(\exists X.\mathcal{P}^{\uparrow}_{\tau_1\ldots\tau_i})[X/X'] \wedge \varepsilon \neq \mathit{false} \implies (\exists X.\mathcal{P}^{\uparrow}_{\tau_1\ldots\tau_i})[X/X'] \wedge \phi_1 \neq \mathit{false}$. **then**
19: **return** SAT and the path $\pi_1\pi_2$
20: **else**
21: /* Refinement */
22: $\mathcal{P} := \mathcal{P} \cup \bigcup_{i\in\{k,\ldots,l\}} \mathcal{P}^{\uparrow}_{\tau_i\ldots\tau_l}$
23: **return** UNSAT

Fig. 1. Algorithm of CEGAR for Non-Zeno Fairness $\exists^{[a_1,\ldots,a_m]}_{\langle b_1,\ldots,b_n\rangle}\Box^{\varepsilon}\phi_1$

VI. Conclusion

In this paper, we presented transition predicate abstraction and CEGAR verification algorithm for a linear hybrid automaton. We are working on implementation of the algorithm. Future work will focus on further practical experiments and evaluation.

VII. Acknowledgement

I wish to thank my advisor Professor Satoshi Yamane, for his continuous support in the project. He was always there to listen and to give advice.

References

[1] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," *Lecture Notes in Computer Science*, vol. 736, pp. 209–229, 1993.

[2] R. Alur, T. Dang, and F. Ivancic, "Reachability analysis of hybrid systems via predicate abstraction," *Lecture Notes in Computer Science*, vol. 2289, pp. 35–48, 2002.

[3] R. Alur and T. Dang, "Counter-Example Guided Predicate Abstraction of Hybrid Systems," *Lecture Notes in Computer Science*, vol. 2619, pp. 208–223, 2003.

[4] F. Wang, "Model-checking distributed real-time systems with states, events, and multiple fairness assumptions," *Lecture Notes in Computer Science*, vol. 3116, pp. 553–568, 2004.

[5] R. Kurki-Suonio, "Hybrid models with fairness and distributed clocks." *Lecture Notes in Computer Science*, vol. 736, pp. 103–120, 1992.

[6] A. Podelski and A. Rybalchenko, "Transition predicate abstraction and fair termination," *Proc. POPL*, vol. 40, no. 1, pp. 132–144, 2005.

[7] B. Cook, A. Podelski, and A. Rybalchenko, "Abstraction refinement for termination," *Lecture Notes in Computer Science*, vol. 3672, pp. 87–101, 2005.