Active Learning : From Blackbox To Timed Connectors

Yi Li* and Meng Sun*
*DI, School of Mathematical Sciences, Peking University, Beijing, China
liyi_math@pku.edu.cn, summeng@math.pku.edu.cn

***Abstract*—Coordination is becoming more and more important, especially in concurrent programming and distributed systems. More and more tools are built to verify coordination models while most of them are facing the same problem: *lack of models*. In many cases, somehow we need to check a connector with nothing more than a binary file. In this paper, we present a learning-based framework to formally verify a Reo connector in case there is no source code. From some practical cases, we've shown the effciency of our approach. Both the algorithm and Reo itself have been ported to *Golang*, making it fully-prepared for parallel programming.**

***Index Terms*—active learning, coordination, formal method.**

I. Introduction

In the past few years, researchers have been focusing on this area, and come up with a series of impressive works. However, most of these works are based on models, instead of binaries. Then it comes a well-known problem: *how can we obtain these models?*

The application of machine learning in formal models has a long history. There are several main orientations: extraction of formal models and learning-guided verification technique.

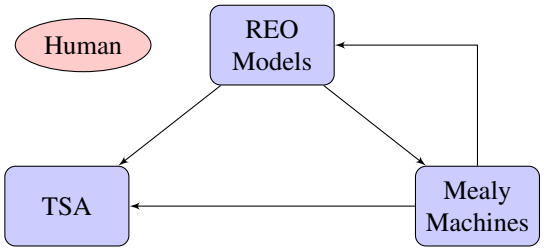

Fig. 1. Our Idea

In this paper, we presented an adapted active learning algorithm to extract timed Reo connectors from binaries with no source code needed.

II. Preliminaries

A. Reo Coordination Language

B. Active Learning and Mealy Machines

III. Timed Connectors as Mealy Machines

Reo Coordination Model, obviously, is not a *proper model* for learning.

So far as we can tell, most works[2] on automata learning are not capable of infinite models. Considering the semantics defined in section II-A, it's apparent that every finite connector has a corresponding constraint automata, and the automata is also finite. Unfortunately, when checking the further definition of Reo connectors' input, we find that it's not the case.

Fig. 2. Infinite States in a Reo Model

While focusing on behavior of connectors, Reo doesn't give detail depiction on behavior of components. As shown in Fig.2, connectors are able to reject any datum if they are not ready. But what will happen outside the connector, if the datum is rejected? This question deserves careful consideration if we are taking an external view.

IV. From Blackbox to Timed Connectors

A. Equivalence Query

Generally, equivalence query has been proved impossible in blackbox models[1]. However, in this section, we're showing that in certain circumstances, equivalence query can be implemented with no approximation.

Since equivalence queries are used to search for counter-examples, firstly let's see what makes counter-examples even existing after the *Table Close* algorithm.

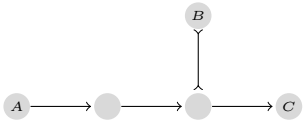

Fig. 3. ¡+caption text+¿

¡++¿

V. Case Studies

VI. Conclusion and Future Work

References

[1] D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.

[2] H. Raffelt and B. Steffen. Learnlib: A library for automata learning and experimentation. In L. Baresi and R. Heckel, editors, *Fundamental Approaches to Software Engineering, 9th International Conference, FASE 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 27-28, 2006, Proceedings*, volume 3922 of *Lecture Notes in Computer Science*, pages 377–380. Springer, 2006.

TODO LIST