Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 11163–11168

Using plant model features to generate reduced test cases for programmable controllers

Canlong Ma, Julien Provost

Technical University of Munich, Safe Embedded Systems, 85748 Garching bei München, Germany (e-mail: {ma, provost}@ses.mw.tum.de).

Abstract: Complete conformance testing is a model-based test technique for programmable controllers. It checks whether an implementation conforms to its specifications with regard to all possible combinations of input signals, which is useful for small scale and safety critical systems. However, the *state space explosion* issue limits its application to large scale systems. This paper presents a method for reducing state space in generation of test cases by utilizing not only specification models but also features extracted from plant models. The application on a benchmark case study shows that the number of test cases is reduced significantly.

Keywords: discrete event system, programmable controller, testing, validation

1. INTRODUCTION

Industrial automation systems such as manufacturing are facing challenges of rising complexity and flexibility. This fact leads to an urgent demand of qualitative testing with convincing reliability and high efficiency (Rösch et al. (2015)). Controllers are key parts in an automation system as they make orders to actuators based on the signals received from sensors. An important task in testing of an industrial automation system is therefore the testing of its controllers.

Programmable controllers with cyclic execution mode are widely used in industry since they are robust enough to endure in industrial environment, have standard programming languages, and meet the 'hard' real-time requirement. In the design phase, specifications are made according to users' requirements. Then, based on the specifications, executable programs are implemented on programmable controllers. In the execution phase, when a programmable controller is turned on, it runs in cycles: reading values of input signals, executing implemented programs, updating values of output signals.

Conformance testing is a model-based test technique that aims to check whether an implementation performs the same behavior with regard to its specifications (Provost et al. (2014)). It is recommended by a series of international certification standards such as *IEC 61850-10* and *IEC 60880*. Conformance testing on a programmable controller consists of three phases: test case generation, test execution, and result verdict.

In the classic strategy, a complete set of test cases is generated from specification models directly and considers all possible combinations of input signals (Provost et al. (2011)). It is worth noting that plant behaviors are usually not considered because they often do not even exist. The few exceptions are: Supervisory control theory (Ramadge and Wonham (1987)), which is unfortunately not receiving the expected attention in the industry up to now; formal verification techniques, some of which use plant models when checking logical properties (Frey and Litz (2000), Machado et al. (2006)); and hardware-in-the-loop test benches (Gu et al. (2007)), which permits to simulate the future plant in a test bench during test execution. Yet, to the best of our knowledge, plant models have not been considered for the generation of test sequences.

Complete conformance testing is advantageous for a *system under test* (SUT) that is safety critical, since it, by design, covers the whole behavior defined by the specification. However, it suffers two issues.

The first issue occurs during test execution and result verdict: erroneous test verdicts might happen due to incorrect detection of synchronous input changes by a programmable controller under test. This has been formally described as single-input-change (SIC) testability issue in Provost et al. (2014). To fulfill the requirement of full SIC-testability in testing programmable controllers, a design-to-test (DTT) approach was proposed in Ma and Provost (2015), which also improves controllability and observability performance in testing additionally. In brief, the DTT approach modifies specification models to improve the testability of their implementations with limited design and testing overhead. For more details, a software toolbox has been developed for the DTT approach (Ma and Provost (2016)).

The second issue is the *state space explosion* issue. With complete testing, the number of test cases grows exponentially with the number of inputs of a SUT, which severely restricts its application to large scale systems.

The method proposed in this paper aims to solve this second issue. The core idea is to use not only specification but also plant models in generation a smaller set of test

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2017.08.1238

cases. With additional information from plant models as well as interactions between specification and plant models, the reduced test cases focus mainly on the nominal behavior of the system, which is more realistic for large scale systems. Moreover, testing the nominal behavior of a SUT is a prerequisite to the test of its faulty behavior; if its nominal behavior is not correct, testing its faulty behavior would be meaningless. Also, for large scale systems, only a subset of the whole behavior is critical and requires a complete testing.

A big advantage of the proposed method is that it does not necessarily require highly detailed or full plant models. Any piece of plant knowledge can contribute to the reduction of test cases. Therefore, in practice, the obstacle of the *state space explosion* issue in conformance testing can be diminished.

The paper is structured as follows: Section 2 presents the formalism of finite state machines used in specification and plant models. Section 3 presents different description methods of signal relations and the utilization of plant models in test case generation. A benchmark case study is illustrated in section 4. Finally, a discussion of this work is given in the last section.

2. BACKGROUND

2.1 Specification model: Communicating Moore machine with Boolean signals

In this paper, specifications of a system are modeled as a set of Moore machines, a type of finite state machine (FSM), which can communicate with each other via internal variables, adapted from Lee et al. (1996).

Due to simplicity reason and a wide range of applications, Boolean signals are used as inputs and outputs in the illustration of the proposed method. However, the method can also be applicable to general digital signals with a few adaptations.

An important thing to keep in mind is that, compared to event based models, signal based models do not restrict only one change of input values at once (Provost et al. (2011)).

A communicating Moore machine extended with Boolean signals is defined by a 7-tuple $(S, s_{init}, I, C, O, \delta, \lambda)$, where:

- S is a finite set of states
- s_{init} is the initial state, $s_{init} \in S$
- I is a finite set of Boolean input signals
- C is a finite set of internal Boolean communicating variables
- O is a finite set of Boolean output signals
- $\delta : S \times 2^{I+C} \rightarrow S$ is the transition function that maps the current state and Boolean expression, which is made up of input signals and communicating variables, to the next state
- $\lambda : S \rightarrow 2^{O}$ is the output function that maps the states to their corresponding output signals

A Boolean expression used in a transition is denoted as a 'transition guard'. A transition is fired when its source state is active and its guard is evaluated as '1' (i.e. *True*). Moore machines are also represented in graphical form in this paper. A simple example is given in Fig. 1.

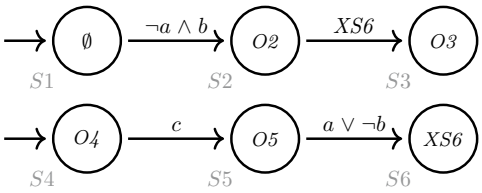

Fig. 1. A simple example of a Moore machine with Boolean signals

A state s is drawn as a circle or a rounded rectangle. A transition δ is represented by an oriented arc with its guard, e.g. $\neg a \wedge b$ for the transition from $S1$ to $S2$.

A state can either have an externally observable action, e.g. *O2* in $S2$, or no observable action, e.g. $\emptyset$ in $S1$. Additionally, a state can also be given an internal communicating variable, e.g. *XS6* in $S6$, which can be used in Boolean transition guards. For example, when the state $S6$ is activated, *XS6* is then assigned the value '1'. If $S2$ is active at the same time, then the transition from $S2$ to $S3$ can be fired.

In hierarchical modeling, a state can contain other states which are called sub-states (Girault et al. (1999)).

2.2 Synchronous composition of individual specification models

When modeling a complex industrial process, it is convenient to build several simple individual models and compose them, instead of directly constructing a large monolithic model.

For composition of FSM models, a significant number of theory research has been done since many years. Practical tools such as Teloco (Provost et al. (2011)) are available to obtain a composed model from individual models.

The formalism of a composed machine is similar to an individual Moore machine. The main differences are:

- In a composed model, a location represents a combination of states from the individual models.
- A transition function between locations is named an 'evolution'.

It is worth mentioning that with Teloco, the composed model contains only stable locations, i.e. locations where only a change in the input values can trigger a change of locations (Provost et al. (2011)).

2.3 Plant model as finite state machine

In this paper, plants can also be modeled as finite state machines with Boolean signals to describe dependency relations between signals.

The formalism of a plant model is similar to a specification model except two terms:

- $\lambda : S \rightarrow 2^{I}$: *inputs* of specification models are used as *outputs* in plant models