Model-based Testing of Cooperating Robotic Systems using Coloured Petri Nets

Raimar Lill, Francesca Saglietti

Model-based Testing of Cooperating Robotic Systems using Coloured Petri Nets

Raimar Lill, Francesca Saglietti

Chair of Software Engineering
University of Erlangen-Nuremberg
Martensstr. 3
91058 Erlangen, Germany
raimar.lill@informatik.uni-erlangen.de
saglietti@informatik.uni-erlangen.de

Abstract. This article proposes a model-based testing approach for cooperating robotic systems. Coloured Petri Nets are used for capturing the high behavioural multiplicity of such systems in a compact and scalable way. For the purpose of systematically extracting test cases from underlying models, a number of coverage criteria based on different model entities is introduced. Finally, in order to ensure practicality, an incremental testing procedure based on increasingly refined coverage concepts is proposed.

Keywords: model-based testing, cooperating autonomous systems, Coloured Petri Nets, coverage metrics, incremental testing procedure

1 Introduction

Reliable cooperation of autonomous, programmable systems assumes that each entity was developed such as to include individual capabilities for sensing its environment, reasoning about the options it offers and deciding for one among several optional behaviours in order to maintain a safe co-existence with the other entities and to contribute to the achievement of a common target.

In contrast to conventional distributed sub-systems subject to a central controller continuously allocating decoupled jobs to each of the communicating partners, modern robotic systems are characterized by high degrees of individual autonomy [1]; this is envisaged for the purpose of increasing flexibility, performance and robustness, in particular by permitting to react quickly and appropriately to unforeseen situations by immediate local intervention.

While certainly adding to the attractiveness of future robotic-based applications, this vision admittedly suffers from a number of limitations, especially concerning the reliability and safety of cooperative behaviour under any operational circumstance. In fact, the flexibility and the agility announced by highly autonomous systems may reveal as risky if the huge width of potential asynchronous behaviour may allow for

sporadic unsafe interactions. It is therefore required to exclude inacceptable side effects by a thorough analysis of all possible interactions.

Due to the physical multiplicity of interacting scenarios, exhaustive static verification is not practicable; a preliminary model-based behavioural representation, however, provides useful support to successive testing phases based on stepwise refined coverage criteria.

For this purpose, the present article deals with testing of cooperative autonomous systems by means of model-based coverage criteria. It is organized as follows: Chapter 2 highlights the main peculiarities of the Coloured Petri Nets modelling language and justifies its choice for the following considerations. Chapter 3 introduces a number of increasingly refined coverage concepts and organizes them in a common subsumption hierarchy. Finally, Chapter 4 suggests how to make use of these coverage concepts in the course of successive testing phases based on increasingly enriched contextual views of operational context.

2 CPN for Modelling Cooperating Autonomous Systems

Coloured Petri Nets (CPN) [2] is a formal discrete-event language for modelling and validating complex concurrent systems. In particular, CPN has proven to be adequate for capturing the high behavioural multiplicity of cooperating autonomous systems thanks to its inherent scalability and expressiveness [3]. The following remarks outline substantial concepts and benefits of CPN, while for a detailed definition of the modelling language the reader is kindly referred to [2] or [3].

Similarly to classical *Place/Transition Petri Nets* [4], also CPN graphs are directed bipartite graphs containing *place* and *transition* nodes, where edges are denoted as *input arcs* if they connect a place with a transition and as *output arcs* if they connect a transition with a place. Contrary to classical Petri Nets with generic tokens, however, CPN tokens are assigned type-specific values named *colours*; more precisely, each CPN place is assigned a type called *colour set* consisting of all different colours tokens may assume in that place. Colour set assignments as well as annotations of arcs by expressions and of transitions by guards are expressed in the functional programming language SML [5].

A CPN transition is enabled as soon as each variable contained in any of its input arc expressions can be bound to a colour of the corresponding input place, such that w.r.t. this variable binding all input places contain at least as many tokens as determined by evaluating the corresponding input arc expression, while satisfying any potential guard of the transition considered. Firing a transition enabled by a given variable binding results in the consumption of input place tokens in quantity and colour determined by corresponding input arc expressions, followed by the production of output place tokens in quantity and colour determined by corresponding output arc expressions.

Concerning the modelling of cooperating robotic systems, CPN provides substantial benefits by offering a relatively intuitive graphical representation supporting visualization by simulation. At the same time, CPN offers the appropriate expressive power by being capable of capturing the high variety of potential interactions arising by cooperation of the robotic entities involved. Thanks to the concept of colour sets, CPN allows to store relevant information in type-specific tokens, thus avoiding the need to expand the graph by additional places and transitions. Hereby it supports not only compactness, but in particular scalability by permitting to adapt to different application-specific conditions.

An example outlining the benefits of CPN was presented in [6]. It is intended to represent the movement of forklifts within a logistic warehouse, where an arbitrary number of robots move along a narrow lane consisting of an arbitrary number of segments for the purpose of accomplishing loading missions. Hereby, decisions on the behaviour of each robot have to be taken as autonomously as possible. The resulting CPN model allows to abstract both from the subjects (robots) and their environment (segments). In particular, it offers a topology-neutral representation which can be easily adapted to plant-specific environmental conditions just by varying the initial CPN markings according to the actual numbers of robots and segments. Generic actions are classified (e.g. robot forward movement, switching positions with a facing robot, alarming a human operator) and represented by *transitions*. On the other hand, action instances referring to specific robots and to specific segments are captured by so-called CPN *events*, i.e. transitions with enabling variable bindings. Further refinement is provided by specifying the global context in which a specific action is carried out. This information is captured by so-called CPN *states*, encoded by net markings allowing for the inclusion of further forklifts not directly involved in the action considered, but potentially interfering in future.

3 Coverage Criteria and Subsumption Hierarchy

According to the principles of model-based testing, test cases are to be derived systematically from underlying CPN models. A CPN-based test case is defined as a pair consisting of an initial state and a finite sequence of events. For the purpose of providing adequate stopping rules to the testing phase, this section introduces coverage metrics based on transition, event and state entities. The following criteria were partly inspired by [7] and [8] where transition-based and state-based coverage criteria for so-called Predicate/Transition Petri Nets [9] were defined. Different classes of coverage criteria for CPN were identified as follows:

Transition-based coverage criteria demand for the execution of basic actions resp. sequences of basic actions; we distinguish between

- the "*all transitions*"- criterion requiring every transition to be fired;
- the "*all transition pairs*"- criterion requiring every transition as well as any possible pair of consecutive transitions to be fired;
- the "*all transition sequences*"- criterion requiring any possible sequence of transitions to be fired.

By restricting the observation to generic action classes, the transition-based coverage concept does not require to take into account the colour of tokens involved in transition firings.

Event-based coverage criteria demand for the occurrence of events, i.e. specific action instances, resp. sequences of events; we distinguish between

- the "*all events*"- criterion requiring every event to occur;
- the "*all event pairs*"- criterion requiring every event as well as any possible pair of consecutive events to occur;
- the "*all event sequences*"- criterion requiring any possible sequence of events to occur.

These criteria explicitly consider the colours and amount of tokens involved in transition firings. On the other hand, they intentionally miss global information outside the narrow scope of the actions addressed.

State-based coverage criteria demand for the reaching of states, i.e. CPN markings, resp. sequences of states; we distinguish between

- the "*all states*"- criterion requiring every state to be reached;
- the "*all state pairs*"- criterion requiring every state as well as any possible pair of consecutive states to be reached;
- the "*all state sequences*"- criterion requiring any possible sequence of states to be reached.

State-based coverage criteria capture global information about operational pre- and post-conditions potentially encountered before and after an event occurrence**.**

Assuming a common initial CPN marking, the coverage criteria introduced above can be organized in the subsumption hierarchy shown in Figure 1. Implications between criteria are indicated by arrows pointing from a stronger to a weaker criterion. For a more detailed explanation of the relations between the coverage criteria proposed, the reader is kindly referred to [6].

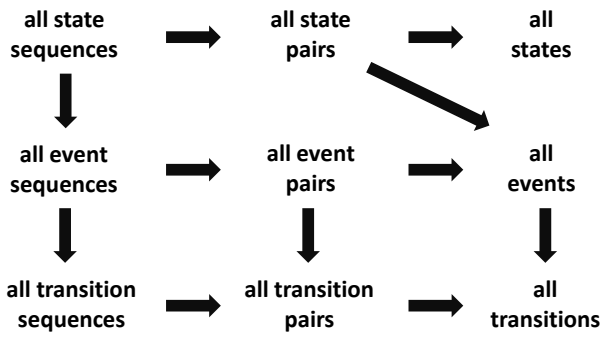

Fig. 1. Subsumption hierarchy of CPN-based coverage criteria

4 Incremental Testing Procedure

The subsumption hierarchy shown in Figure 1 illustrates a fundamental refinement relation between criteria, as stronger coverage concepts placed in the upper part of the picture result from weaker ones (shown in the bottom part) by including additional contextual details. This insight suggests to organize the overall testing process in an incremental fashion by starting with a narrow testing focus restricted to generic actions and by gradually enriching the scope of behavioural variety with further contextual information. The resulting procedure is summarized in Figure 2.

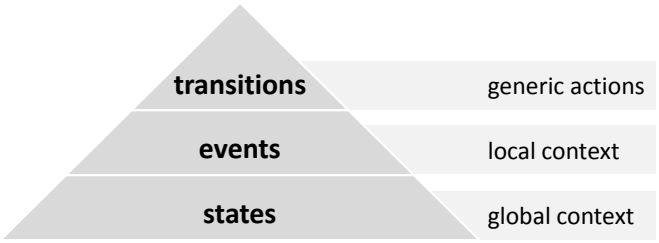

Fig. 2. CPN entities involving different degrees of contextual detail

Referring to the cooperating forklifts presented as example in Section 3, generic action testing aims at verifying basic, but non-trivial functionalities like sensing, self-localisation and motor activities of robots. The following testing phase addresses local context by adding further contextual information on specific action instances like movement on segments of varying grip or slope. As a further refinement step, a third and final structural testing phase is intended to capture also global states encountered before and after any event occurrence. Covering consecutive state pairs does not only address specific action instances, but also global environmental conditions. In case of the forklift factory, state pairs may capture the information on the positioning of any other robot not directly involved in the action under test, but potentially affecting its

future behaviour, e.g. a traffic jam to be encountered after the switching of positions of two facing robots.

The three testing phases mentioned so far are characterized by systematic, structural demands ignoring the frequency of occurrence of individual events under given circumstances. Obviously, full coverage may only be achieved up to practicable sequence lengths. Nonetheless, this testing process is considered as extremely useful to support accurate preliminary verification activities carried out for the purpose of detecting faults; it does not allow, however, to assess operational reliability. For this purpose, systematic testing must be followed by a profile-based field testing phase with state sequences weighted according to their expected operational frequency and safety criticality.

Table 1 summarizes the overall testing procedure we propose.

	Entities to be covered	**Testing objective**
Structural testing	transitions / transition sequences	generic actions
	events / event sequences	local context
	state pairs / state tuples	global context
Field testing	profile-specific state sequences	operational conditions

Table 1. CPN-based testing procedure

5 Conclusion

In this article a step-wise CPN-based testing approach for cooperating robotic systems was introduced. To provide measureable stopping rules to the structural test, different coverage criteria were defined and hierarchically organized in a subsumption hierarchy. The implications between the coverage concepts suggest a traceable structural testing procedure consisting of three major steps:

- *generic action testing* addressing basic actions;
- *local context testing* addressing data-specific action instances;
- *global context testing* addressing state-dependent action instances.

While these systematic testing phases are essential for capturing the multiplicity of interaction scenarios, they must be followed by a fourth phase based on operational profiles supporting reliability assement by statistical testing [10].

Ongoing work is being devoted to automatic test case generation via evolutionary techniques which already proved as successful approaches in more conventional software testing areas [11], [12].

Acknowledgement: It is gratefully acknowledged that part of the work reported was sponsored by the German Federal Ministry of Education and Research BMBF (Bundesministerium für Bildung und Forschung) in cooperation with the European Union Research Programme ARTEMIS (Advanced Research and Technology for Embedded Intelligence and Systems), project R3-COP (Resilient Reasoning Robotic Co-operating Systems).

References

1. Saglietti, F., Söhnlein, S., Lill, R.: Evolution of Verification Techniques by Increasing Autonomy of Cooperating Agents. In: Autonomous Systems Developments and Trends, Studies in Computational Intelligence, vol. 391, pp. 353-362. Springer (2011)
2. Jensen, K., Kristensen, L. M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer (2009)
3. Lill, R., Saglietti, F.: Model-based Testing of Autonomous Systems based on Coloured Petri Nets. In: ARCS 2012 Workshops Proceedings, LNI, vol. 200, pp. 241-250. Gesellschaft für Informatik (2012)
4. Murata, T.: Petri Nets: Properties, Analysis and Applications. In: Proceedings of the IEEE, vol. 77, no. 4, pp. 541-580. IEEE (1989)
5. Milner, R., Tofte, M., Harper, R., MacQueen, D.: The Definition of Standard ML (Revised). MIT Press (1997)
6. Lill, R., Saglietti, F.: Test Coverage Criteria for Autonomous Mobile Systems based on Coloured Petri Nets. In: 9th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems (FORMS/FORMAT 2012), pp. 155-162. Institut für Verkehrssicherheit und Automatisierungstechnik, Braunschweig (2012)
7. Zhu, H., He, X.: A Theory of Testing High Level Petri Nets. In: Proc. 16th Int. Conf. on Software - Theory and Practice, pp. 443-450. Publishing House of Electronics Industry (2000)
8. Zhu, H., He, X.: A Methodology of Testing High-Level Petri Nets. In: Information and Software Technology, vol. 44, no. 8, pp. 473-489. Elsevier (2002)
9. Genrich, H. J., Lautenbach, K.: System Modelling with High-Level Petri Nets. In: Theoretical Computer Science, vol. 13, issue 1, pp. 109-136. Elsevier (1981)
10. Söhnlein, S., Saglietti, F., Bitzer, F., Meitner, M., Baryschew, S.: Software Reliability Assessment based on the Evaluation of Operational Experience. In: Proc. 15th Internat. GI/ITG Conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance, LNCS, vol. 5987, pp. 24-38. Springer (2010)
11. Pinte, F., Oster, N., Saglietti, F.: Techniques and Tools for the Automatic Generation of Optimal Test Data at Code, Model and Interface Level. In: Companion of 30th International Conference on Software Engineering (ICSE 2008), pp. 927-928. ACM (2008)
12. Meitner, M., Saglietti, F.: Software Reliability Testing Covering Subsystem Interactions. In: Proc. 16th International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB & DFT 2012), LNCS, vol. 7201, pp. 46-60. Springer (2012)