9th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems
Institut für Verkehrssicherheit und Automatisierungstechnik, TU Braunschweig, 2012
FORMS/FORMAT 2012 (http://www.forms-format.de)

Test Coverage Criteria for Autonomous Mobile Systems based on 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. For the purpose of testing the cooperative behavior of software-controlled autonomous systems, a model-based testing approach is presented. It makes use of Coloured Petri Nets providing a compact and scalable representation of behavioral multiplicity to be covered by an appropriate selection of representative test scenarios fulfilling net-based coverage criteria.

1 Introduction

For reasons of performance and flexibility, software-controlled mobile systems aim at a high level of autonomy, while cooperating for the purpose of carrying out common tasks or of resolving common conflicts. The degree of decisional freedom left to each individual vehicle is maximized, subject to synchronization points required to ensure proper interactions when necessary. For this reason, the concurrent movement of autonomous systems usually gives rise to high behavioral multiplicity, typically growing exponentially with the number of moving entities involved.

Evidently, such an increasing degree of complexity has crucial consequences for software testing: in order to ensure appropriate degrees of behavioral representativeness, test selection has to be systematized such as to allow for objectively reproducible coverage measures. The approach chosen to identify adequate test scenarios capturing as much interaction multiplicity as reasonably possible relies on

– modeling the behavior of autonomous systems by Coloured Petri Nets (s. chapter 2),
– defining coverage criteria based on graphical entities of Coloured Petri Nets (s. chapter 3),
– generating test scenarios fulfilling these criteria (s. chapter 4).

2 Model-based Testing for Coloured Petri Nets

Coloured Petri Nets (CPN) [1] provide a well-known notation for modeling and analyzing complex concurrent and distributed systems. In particular, they have proven to support the representation of high-scale autonomous systems thanks to their scalability [2]. For a detailed definition, we refer to [1] and restrain the following introductory remarks to very basic considerations.

As generally known, CPN enrich ordinary Place/Transition Petri Nets [3] by allowing for value-specific tokens (*colours*). Furthermore, for the purpose of controlling the consumption and production of tokens, CPN arcs are annotated by expressions, while CPN transitions may be annotated by guards. More precisely, in order for a transition to fire, each variable occurring along an input arc is associated with a colour of the corresponding input place (*variable binding*), such that tokens of this colour are present in sufficient number in the input place considered (as determined by evaluating the arc expression) and all variable values fulfill potential transition guards. By firing the transition, the number of tokens to be consumed in the input places and produced in the output places is determined by evaluating the corresponding input arc and output arc expressions w.r.t. the enabling variable binding.

A transition together with a variable binding enabling its firing is denoted as a CPN event. CPN models are tested by simulating their execution from an initial state given by a specific CPN marking, processing test cases defined as sequences of CPN events.

3 Test Coverage Criteria for Coloured Petri Nets

3.1 Entity-based Classes of CPN Coverage Criteria

In order to provide objectively measurable test stopping rules, appropriate CPN coverage criteria are to be defined. Already existing approaches address the coverage of graphical elements of Petri Nets. Among them, [4] introduces coverage criteria for Prioritized Time Petri Nets based on places, markings and transitions.

The following CPN coverage concepts were inspired by the approaches published in [5] and [6] which were originally focused on Predicate/Transition Petri Nets. They depend on the definition of initial CPN markings modeling potential pre-states. Different classes were identified as follows.

Transition-based coverage criteria address the firing of transitions reflecting the execution of actions. Therefore, this testing strategy is focused on the verification of basic system functionality. In particular, transition-based testing does not take into account the varying colour of CPN tokens, thus neglecting to distinguish between different action instances. The following transition-based coverage criteria may be considered:

- the "*all transitions*"- criterion requires the firing of every transition;
- the "*all transition pairs*"- criterion requires the firing of every transition as well as the consecutive firing of any possible pair of sequential transitions;
- the "*all transition sequences*"- criterion requires the firing of any possible sequence of transitions.

Event-based coverage criteria address the occurrence of (individual or successive) events. In particular, this strategy takes into account colour variety. The following event-based coverage criteria may be considered:

- the "*all events*"- criterion requires the occurrence of every event;
- the "*all event pairs*"- criterion requires the occurrence of every event as well as the consecutive occurrence of any possible pair of sequential events;
- the "*all event sequences*"- criterion requires the occurrence of any possible sequence of events.

State-based coverage criteria address the reaching of (individual or sequential) states, where a state is represented by a CPN marking**.** In addition to event-based information, this testing strategy also takes into account operational conditions before and after an event occurrence. The following state-based coverage criteria may be considered:

- the "*all states*"- criterion requires to cover every state;
- the "*all state pairs*"- criterion requires to cover every state and any possible pair of sequential states;
- the "*all state sequences*"- criterion requires to cover any possible sequence of states.

3.2 Subsumption Hierarchy of Coverage Criteria

In order to exclude syntactically legal, but unrealistic or redundant CPN models, in the following it is assumed that all transitions are connected by arcs, all guards are satisfiable and events leading from one state to another are uniquely defined.

As explained in the following, assuming the same initial net marking the coverage criteria mentioned above can be organized in the subsumption hierarchy shown in Figure 1.

- As event-based coverage criteria require the firing of all possible transitions in combination with all possible variable bindings, they subsume in particular the corresponding transition-based coverage criteria.
- As events trigger state changes, each edge of a CPN state space graph can be associated with an event, where different edges may relate to the same event, but only one event relates to a specific change of state. Therefore, the "all *state sequences*"- criterion evidently subsumes the "*all event sequences*"- criterion.
- The same implication, however, is not necessarily true for state and event pairs, as the coverage of an event pair requires the coverage of the state triple consisting of the initial, intermediate and final state occurring before, between and after the occurrence of the two events.
- On the other hand, the "*all state pairs*"- criterion evidently subsumes the "*all events*"- criterion, as traversing a state pair obviously implies the coverage of the event triggering the corresponding state change.

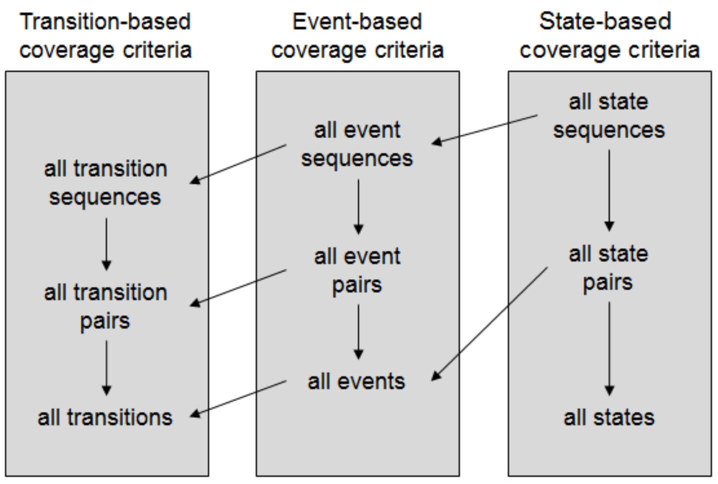


Fig. 1. Subsumption hierarchy of CPN-based coverage criteria

Evidently, test case generation based on CPN events or states relies on the construction of the underlying state space graph with nodes representing states and edges representing events. In case the number of states is prohibitively large or even infinite, the state explosion problem may be circumvented by attempting to reduce the states to be covered to a relevant subset or to constrain the CPN model via domain-specific bounds. Alternatively, one may decide to focus the testing process to more affordable transition-based coverage criteria.

4 Application of CPN-based Test Concepts to a Robot Factory

In this chapter, the concepts introduced above are illustrated by means of an industrial application requiring a number of forklifts to cooperate to a common loading task by moving along segments, sensing obstacles and avoiding conflicts. Depending on the number of forklifts and segments, a high multiplicity of potential interaction scenarios can be captured by the same model, thus demonstrating the scalability of CPN.

This application was modeled using CPN Tools [7]; the resulting CPN is shown in Figure 2, allowing for an arbitrary number of robots (set *RB*) and of segments (set *SEGMENT*). Each robot is assigned a mission represented by a triple (*r*, *s*, *scurr*) where *r* denotes the robot in question, *s* its target segment and *scurr* its present location. Each robot aims at accomplishing its mission as autonomously as possible by

– determining by optical sensors whether the next segment can be accessed or is occupied;
– if the segment is free, the robot will move forward (transition *go left* or *go right* depending on its direction);
– if the segment is occupied by an obstruction (passive obstacle or active robot targeting the same direction), it will try to access the segment for at most five times before triggering an alarm (transition *traffic holdup*), hereby requesting human intervention;
– if the segment is occupied by an active robot targeting the opposite direction, both of them will cooperate by switching their positions (transition *switching maneuver*).

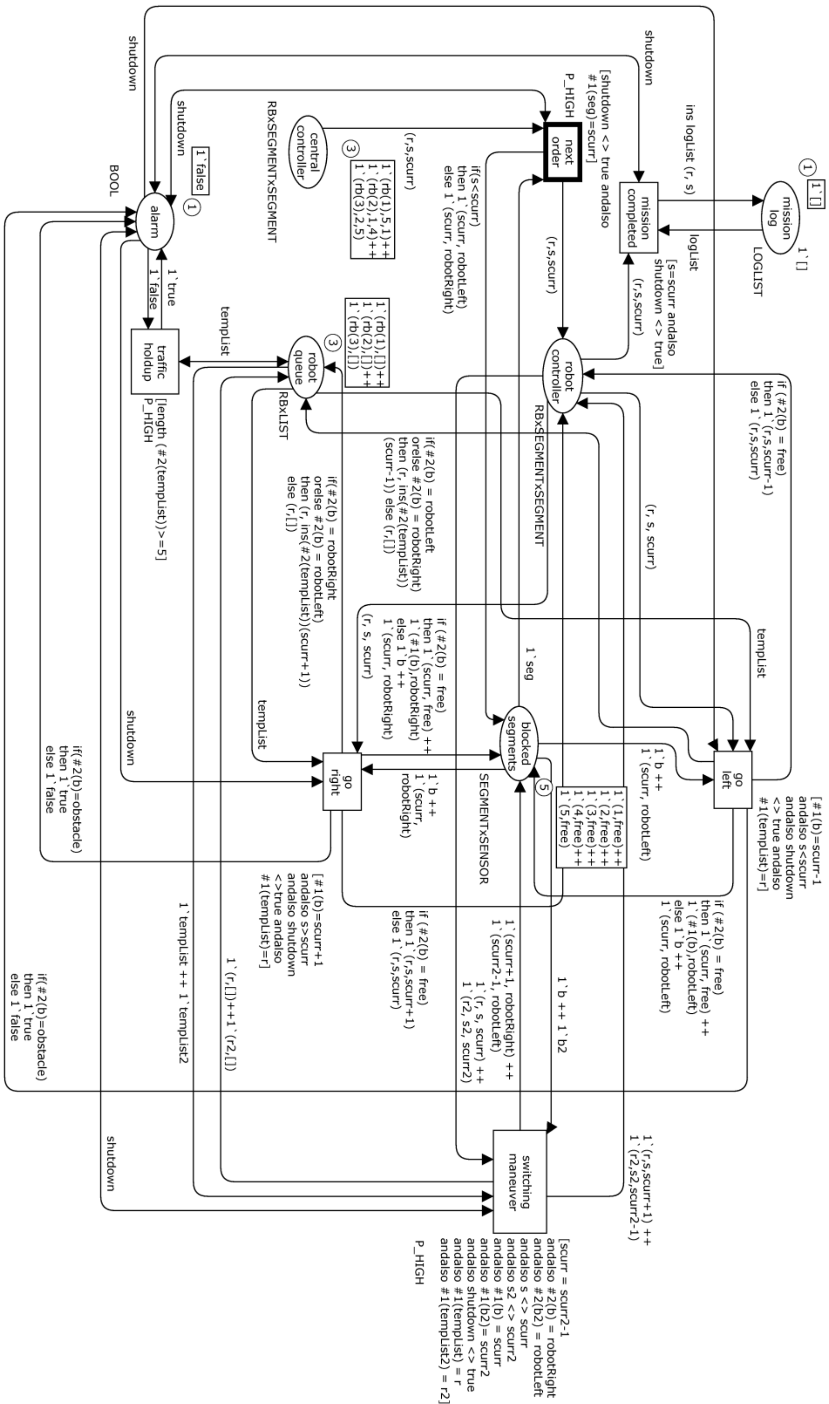
Although the robots behave autonomously, the CPN model keeps track of their location by updating the information on the current segment allocation (place *blocked segments*). It should be noted that global data is not accessible by individual robots whose knowledge is constrained to sensor perception. Similarly, to keep track of successive access retries due to blockades, a list of consecutive retry attempts is managed for every robot and segment (place *robot queue*). To allow for fairness, both transitions *traffic holdup* and *switching maneuver* are assigned high priority (supported by CPN Tools). Completed missions are logged in the order of their accomplishment (place *mission log*).

The initial marking shown in Figure 2 relates to the particular case of 3 robots moving along a lane consisting of 5 segments and processing the following missions:

– robot 1 is initially positioned in segment 1, targeting segment 5 (R#1: 1→5);
– robot 2 is initially positioned in segment 4, targeting segment 1 (R#2: 4→1);
– robot 3 is initially positioned in segment 5, targeting segment 2 (R#3: 5→2).

Table 1 shows the number of CPN entities to be covered for an increasing number of robots (between 2 and 4) moving along 5 segments with individual missions.

For the purpose of test case generation, it is planned to apply evolutionary approaches to multi-objective optimization aiming at maximizing test coverage while minimizing test amount. Similar techniques were successfully developed and applied to unit testing [8], integration testing [9] and reliability testing [10].



mission log
LOGLIST
1`[]
mission completed
ins logList (r, s)
logList
[s=scurr andalso shutdown <> true]
next order
P_HIGH
[shutdown <> true andalso #1(seg)=scurr]
central controller
RBxSEGMENTxSEGMENT
1`(rb(1),5,1)++ 1`(rb(2),1,4)++ 1`(rb(3),2,5)
(r,s,scurr)
if(s<scurr) then 1`(scurr, robotLeft) else 1`(scurr, robotRight)
robot controller
RBxSEGMENTxSEGMENT
robot queue
RBxLIST
1`(rb(1),[])++ 1`(rb(2),[])++ 1`(rb(3),[])
alarm
BOOL
1`false
1`true
traffic holdup
[length (#2(tempList))>=5] P_HIGH
tempList
shutdown
blocked segments
SEGMENTxSENSOR
1`(1,free)++ 1`(2,free)++ 1`(3,free)++ 1`(4,free)++ 1`(5,free)
1`seg
go left
go right
switching maneuver
if (#2(b) = free) then 1`(r,s,scurr-1) else 1`(r,s,scurr)
if (#2(b) = free) then 1`(r,s,scurr+1) else 1`(r,s,scurr)
[#1(b)=scurr-1 andalso s<scurr andalso shutdown <> true andalso #1(tempList)=r]
[#1(b)=scurr+1 andalso s>scurr andalso shutdown <>true andalso #1(tempList)=r]
if(#2(b)=obstacle) then 1`true else 1`false
1`b ++ 1`b2
1`(r,s,scurr+1) ++ 1`(r2,s2,scurr2-1)
1`templist ++ 1`tempList2
1`(r,[])++1`(r2,[])
[scurr = scurr2-1 andalso #2(b) = robotRight andalso #2(b2) = robotLeft andalso s <> scurr andalso s2 <> scurr2 andalso #1(b) = scurr andalso #1(b2)= scurr2 andalso shutdown <> true andalso #1(tempList) = r andalso #1(tempList2) = r2]

Fig. 2. Model of the CPN Robot Factory

number of robots, location and mission / **CPN entities**	2	3	4
	R#1: 1→5 R#2: 5→1	R#1: 1→5 R#2: 4→1 R#3: 5→2	R#1: 1→4 R#2: 2→5 R#3: 4→1 R#4: 5→2
transitions (mission-unspecific)	6	6	6
events	18	72	164
states	33	261	830

Table 1. Number of CPN entities for an increasing number of robots with pre-defined missions

5 Conclusion

This article introduced a number of net-based coverage criteria for mobile systems cooperation modeled by CPN. Implications between different criteria were graphically captured by a subsumption hierarchy. The concepts developed were successively applied to an exemplifying robot factory, in particular highlighting major differences between coverage criteria in terms of the number of entities required to be covered. The systematic and measurable approach proposed in this article will offer the basis for automatic test generation procedures aimed at multi-objective optimization.

Acknowledgement: It is gratefully acknowledged that part of the work reported was sponsored by 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. Jensen, K., Kristensen, L. M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. pp. 95-125. Springer, 2009
2. 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
3. Murata, T.: Petri Nets: Properties, Analysis and Applications. In: Proceedings of the IEEE, vol. 77, no. 4, pp. 542-543. IEEE, 1989
4. Adjir, N., De Saqui-Sannes, P., Rahmouni, K. M.: Testing Real-Time Systems Using TINA. In: Testing of Software and Communication Systems, LNCS, vol. 5826, pp. 1-15. Springer, 2009
5. 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
6. 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
7. Jensen, K., Kristensen, L. M., Wells, L.: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. In: International Journal on Software Tools for Technology Transfer, vol. 9, no. 3-4, pp. 213-254. Springer, 2007
8. Oster, N., Saglietti, F.: Automatic Test Data Generation by Multi-Objective Optimisation. In: Proc. 25th Int. Conf. on Computer Safety, Reliability, and Security, LNCS, vol. 4166, pp. 426-438. Springer, 2006
9. Saglietti, F., Pinte, F.: Automated Unit and Integration Testing for Component-based Software Systems. In: Proc. Workshop on Dependability and Security for Resource Constrained Embedded Systems, article no. 5. ACM Digital Library, 2010
10. Söhnlein, S., Saglietti, F., Bitzer, F., Meitner, M., Baryschew, S.: Software Reliability Assessment based on the Evaluation of Operational Experience. In: Proc. 15th International GI/ITG Conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance, LNCS, vol. 5987, pp. 24-38. Springer, 2010