TRANSFORMATION OF GUARDED STATECHARTS FOR QUANTITATIVE EVALUATION OF DEPENDABLE EMBEDDED SYSTEMS

M. Dal Cin[1], Huszerl G.[2] and K. Kosmidis[1]

We propose a modeling and an evaluation technique for the validation of the dependability of distributed systems such as networked embedded systems that consist of autonomous loosely coupled nodes. The approach is based on the Guarded Statecharts and on a transformation of them to models amenable to a quantitative evaluation, in particular, to Generalized Stochastic Petri Nets. It permits to compound a new modeling technique with proven evaluation tools.

1. Introduction

Statecharts (state diagrams) represent finite state machines. They describe the behavior of objects in response to external stimuli. Thus, statecharts can be used to model the reactive, state-driven system behavior of embedded systems. Many embedded systems are safety and reliability critical. This makes a dependability analysis of their behavior necessary. This analysis deals, for instance, with the mean duration of a recovery cycle or the probability of a failure. For a quantitative dependability analysis it is, however, necessary to model the environment as well as the control software of embedded systems, since the environment can be the source of faults which can give rise to errors in the execution of the control software.

[1]University of Erlangen-Nuremberg (Germany), Department of Computer Structures (IMMD3)
e-mail: dalcin@informatik.uni-erlangen.de, kosmidis@informatik.uni-erlangen.de
[2]Technical University of Budapest (Hungary), Department of Measurement and Information Systems
e-mail: huszerl@mit.bme.hu

In this paper, we define Guarded Statecharts (GSCs), describe the fault model, and outline the transformation of the GSCs to Generalized Stochastic Petri Nets (GSPNs)[1].

2. Guarded Statecharts

Thus, dependability evaluation of embedded systems tends to be very complex causing the modeling problem to be notoriously elusive and error prone. Therefore, when modeling embedded systems a trade-off has to be made between the degree of detail in modeling the possible behaviors of a system and the degree of automation of the analysis process. This necessity lead us to define a sub-class of statecharts comprising so-called Guarded Statecharts.

Guarded Statecharts (GSCs) form a sub-class of statecharts ([2], [3], [4]) which is suited for modeling dependable embedded systems. Embedded systems can be modeled by the AND-composition of concurrent GSCs. The main objects of a GSC are states (container states, base states, initial states) and transitions with guards. In addition, labels of transitions describe timing information (arrival distribution of signals) or static information (probabilities of possible outcomes).

2.1 The Definition

Guarded Statecharts: Given a set Ev of external event-variables, a guarded state chart (GSC) is a finite set A of actions with guards and a finite set S of states one of which is the i-state (initial state) of the GSC. Each state and each event has a name. Actions denote state transition. When state transitions are depicted graphically, they are labeled with labels of the form [*guard*] or *ev*, where '*guard*' is a name of a guard and *ev* is the name of an external event.

-*Guards* are boolean expressions of the predicates *in(<state>)* where *in(<state>)* evaluates to true, if <*state*> is the (actual) i-state of the GSC or of some concurrent GSC.

-*Actions* are of the form:

<guard>*<set_of_states>;or <trigger>*<set_of_states>

For instance, Figure 1 shows the graphical equvivalent of *in*(Table.down) *AND* (*in(*Feed_Belt.move) OR in (Feed_Belt.Stop)*(Feed_Belt-stop)

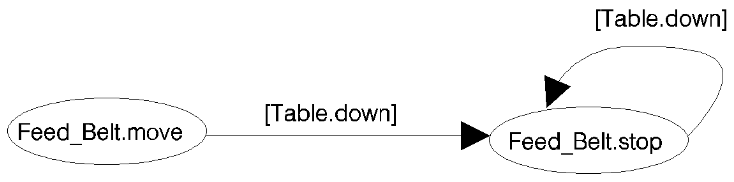

Figure 1: Action

The trigger is a boolean expression of atomar predicates over events and the <guard> is a guard expression. We restrict guards of an action by stipulating that, if a guard contains more than one state of S, the predicates of these states are OR-connected. The action is such that it is executed atomically and instantaneously, if its trigger or its guard evaluates to true. The effect of the execution is that exactly one state of <set_of_states> is chosen non-deterministically as next i-state of the GSC.

2.2 The Semantics

Outputs are considered to be part of the state in which they occur. Guards can be considered as high-level abstractions of synchronization mechanisms. Using GSCs we can abstract continuous signals to discrete signals assuming a finite set of critical values. For example, it is only important to observe whether a robot arm is directed in a position allowing for unloading, or pointing toward a press; all intermediate positions can be collapsed into a single third value.

This way, we model sensors and actuators via states. A state representing an actuator being active means that the actuator is set; analogously if a component is in a state which represents a sensor, it means that this sensor is set. Moreover, in GSCs hardware and software components are only allowed to communicate via such sensor and actuator states.

This interaction is expressed via guard expressions containing predicates over sensor or actuator states. Likewise, interactions between tasks of the control software are also modeled by guarded state transitions: this corresponds to an asynchronous synchronization pattern between tasks. This pattern is inherently multithreaded, because it models a message being passed to another object without the yielding of control.

3. The Fault Model

For the dependability analysis, faults, errors and failures are modeled by message losses, loss of synchrony, erroneous states, and erroneous state transitions. Many of these faults and errors can be modeled by so-called state perturbations. State perturbations include distinguished states corresponding to degraded performance of the modeled system, path leading to those states, erroneous state transitions, trigger events due to external faults giving rise to erroneous state transitions and the use of guards to express fault-tree like failure conditions. Thus, a wide spectrum of possible errors can be modeled by state perturbations. Particularly, state perturbations are adequate modeling techniques when dealing with guarded statecharts.

An other type of state perturbations arises, if guards of a Guarded Statecharts are not observed. For example, the guard *in*(*press.up*) may not be observed by the robot; that is, this guard always evaluates to true. This way, sensor and actuator faults can easily be modeled. Finally, using guards fault trees over component states can be integrated into statecharts. As an example see Figure 2.

Figure 2: Fault Tree

4. The Transformation

For a dependability analysis GSC-models must be transformed to models amenable to mathematical analysis. Guarded Statechart can directly be transformed to Generalized Stochastic Petri Net. State transitions with time delay are transformed to timed Petri Net transitions, those without time delay to immediate Petri Net transitions. Guards become guards of Petri Net transitions. For instance, in PANDA, our Petri net tool, it is possible to annotate transitions with guards and to use state dependent capacities for arcs.

Figures 3 and 4 show a simple example: two interacting statecharts (modeling a device and its control) exchanging sensor and actuator signals (via sensor and actuator states) and their transformation to Generalized Stochastic Petri nets.

Figure 3: GSCs

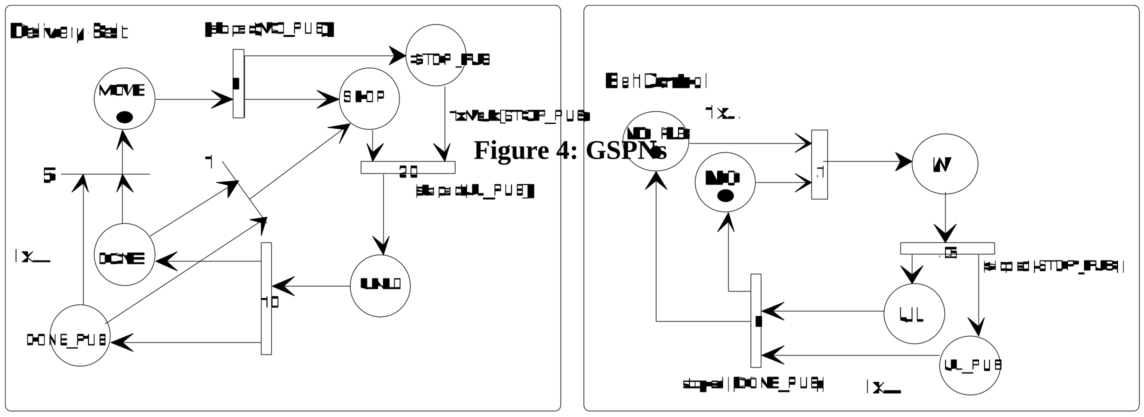

Figure 4: GSPNs

References

[1] M. Ajmone Marsan, et. al.: PeformanceModels of Multiprocessor Systems, The MIT Press 1986
[2] D. Harel, A. Naamad: The STATEMATE Semantics os Statecharts,

ACM Trans. Soft. Eng. Method. 5:4 (Oct. 1996), October 1995; revised July 1996

[3] D. Harel, A. Pnueli, J. P. Schmidt, R. Sherman: On the Formal Semantics os Statecharts (extended abstract), Proceedings Symposium on Logic in computer Science, pages 54-64, Ithaca, New York,

June 22-25 1987, IEEE Computer Society Press

[4] G. Booch, I. Jacobson and J. Rumbaugh: The Unified Modeling Language, User Guide,

Addison Wesley 1999