Electronic Communications of the EASST
Volume X (2010)

Proceedings of the 10th International Workshop on Automated Verification of Critical Systems (AVoCS 2010)

Design-Time Model Checking in Part-Whole Statecharts

Luca Pazzi, Marco Pradelli, Matteo Interlandi

3 pages

Guest Editors: Jens Bendisposto, Michael Leuschel, Markus Roggenbach
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Design-Time Model Checking in Part-Whole Statecharts

Luca Pazzi, Marco Pradelli, Matteo Interlandi

University of Modena and Reggio Emilia
Via Vignolese 905, Modena, Italy
luca.pazzi@unimore.it

Abstract: We present recent research concerning model checking applied to an extension of Statecharts by which it becomes possible at design time to incrementally compose already verified modules into higher-level assemblies.

Keywords: modularity, compositionality, design-time model checking

1 Extended Abstract

By the traditional Statecharts coordination and communication model it is possible to assemble a global behavior from component behaviors hosted in separate concurrent modules, by letting the state machine residing on each component module to act on each other. For example, two concurrent Statecharts sections, each hosting the behavior of a traffic light, will synchronize by triggering a transition into a concurrent state machine, which in turn will propagate further events to the parallel section in order to keep the companion traffic light synchronized. Specifying and checking formal properties through such a model is usually done by model checking techniques where the global states of the system are arranged along a tree. Such techniques present the disadvantage of being loosely related to the design phase and of becoming potentially ineffective as the number of component behaviors grows.

Part-Whole Statecharts are instead composed by two main sections and have a different communication model [Paz08]. The former section, called *assembly section*, hosts the set of component behaviors, which are not allowed to communicate one with the other and are therefore deprived of mutual references: coordination among them is instead obtained through a state machine hosted in the latter section, called *whole section*. Such a state machine allows the designer to explicitly enforce the desired global behavior, as shown in Figure 1-(a), where the switchover among the two traffic lights is obtained by moving from state Main_Enabled to state Sec(ondary)_Enabled and vice versa, each state denoting one of the two crossing roads enabled. The global behavior is obtained through list of events directed towards the assembly of components, which label state transitions in the whole section and are called *actions*. For example transition t_2 in Figure 1-(a) triggers traffic light m from R(ed) to G(reen) through action m.go and traffic light s from G to R through action s.stop. The state machine in the whole section acts as an interface for further composition.

In a recent research under final refinement [Paz08], we describe a formal method which allows to enforce, *at design time*, a formal meaning to each state S in the whole section, through a logical proposition, called *state invariant*. Such an invariant denotes the set of global states the components behaviors are allowed to assume when the control is in state S. For example state Main_Enabled is associated to invariant proposition $\mathbf{I}_1 = \mathsf{m.G} \wedge \mathsf{s.R}$ – meaning that the two

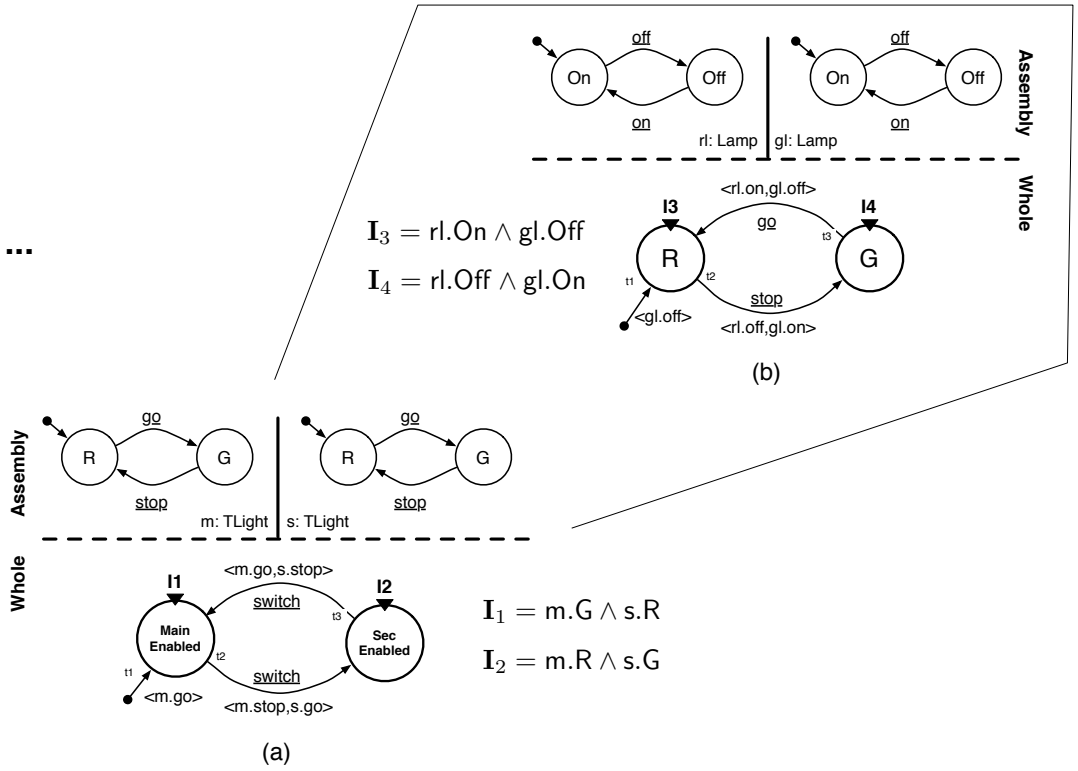

Figure 1: A crossroad behavior (a) is assembled by two traffic light behaviors m and s, which are in turn assembled by two basic lamp behaviors rl and gl (b). State invariants are pictorially associated with states in the whole section by black triangles.

traffics lights have to be, respectively, red and green. Vice versa, state Sec_Enabled is associated to invariant proposition $\mathbf{I}_2 = \mathsf{m.R} \wedge \mathsf{s.G}$ – meaning that the two traffics lights have to be, respectively, green and red. The method works by verifying, statically, that state invariants hold by checking that

1. incoming transitions to a generic state S do not violate the invariant of S;

2. autonomous, i.e. non controllable, transitions (for example being part of the behavior of a sensor device) which may invalidate the invariant $\mathbf{I}$ of state T, are properly handled.

The first check is accomplished by computing, for each state transition t from state S to T, a *transition postcondition* $\mathsf{p}(t)$, which holds of the global state of the assembly after t takes place. Proposition $\mathsf{p}(t)$ must be implied by $\mathbf{I}(T)$, the invariant of the transition arrival state T. The state invariant must also be equal to the logical sum of the incoming transition postconditions, that is $\mathbf{I}(S) = \vee_{t \in T(S)} \mathsf{p}(t)$, where $T(S)$ is the set of state transitions which have S as final state.

Proposition $\mathsf{p}(t)$ is computed by assuming that the assembly of components satisfies $\mathbf{I}(S)$, the invariant of the transition start state T, before t takes place and that each action which is

associated with t modifies it incrementally resulting in $\mathsf{p}(t)$. For example $\mathbf{I}_1 = \mathsf{m.G} \wedge \mathsf{s.R}$ becomes $\mathbf{I}'_1 = \mathsf{m.R} \wedge \mathsf{s.R}$ after action m.stop takes place, and so on. After performing n actions $\langle a_1, \ldots, a_n \rangle$, the starting state invariant $\mathbf{I}_1$ is transformed into $\mathbf{I}_1^{(n)} = \mathsf{p}(t)$, with a semantics akin to that of *generalized substitutions* in the B-Method [Abr96], restricted to assignment and sequential composition of actions; we are investigating further similarities.

In order to accomplish the second check, we developed an algorithm which is able to determine, in advance, the set of all the couples (δ, c), named *exit zones*, where δ is an autonomous state transition whose happening may invalidate $\mathbf{I}$ when the system satisfies condition c. Specific transitions departing from T which react to the happening of δ and are guarded by c, have then be introduced with the aim of bringing the system to a new state complying with a valid state invariant.

Once state invariants are enforced by the method described above, verification of safety and liveness properties, such as "both roads are never enabled at the same time" for the system of Figure 1-(a), can be finally achieved by observing that, since the system control moves only within the states of the whole section, either $\mathbf{I}_1$ or $\mathbf{I}_2$, but not both, will hold at any time.

We emphasize that verification is also *compositional*, since each component behavior is already verified with respect to its own safety and liveness properties. For example, in Figure 1-(b) the behavior of a traffic light is built from the behavior of component lamps rl and gl, and satisfies its own state invariants, $\mathbf{I}_3 = \mathsf{rl.On} \wedge \mathsf{gl.Off}$, associated to state R, and $\mathbf{I}_4 = \mathsf{rl.Off} \wedge \mathsf{gl.On}$, associated to state G. Following modular composition, higher level invariants imply lower level ones: for example, stating that a "road is stopped" takes for granted that "the red lamp is lit and the green one is dimmed" without having to check the latter proposition each time the whole system is assembled from its parts.

Bibliography

[Abr96] J.-R. Abrial. *The B-book: assigning programs to meanings*. Cambridge University Press, New York, NY, USA, 1996.

[Paz08] L. Pazzi. Modular Model Checking in Part-Whole Statecharts. Technical report CRIS-2008-02-01, 2008. http://cris.unimore.it/cris/files/CRIS-2008-02-01.pdf.