Comfortable Modeling of Complex Reactive Systems

Steffen Prochnow and Reinhard von Hanxleden
Real-Time and Embedded Systems Group, Dept. of Computer Science and Applied Mathematics
Christian-Albrechts-Universität of Kiel, Olshausenstr. 40, D-24118 Kiel, Germany
{spr, rvh}@informatik.uni-kiel.de

Abstract

Modeling systems based on semi-formal graphical formalisms, such as Statecharts, has become standard practice in the design of reactive embedded devices. However, the modeling of realistic applications often results in very large and unmanageable graphics, severely compromising their readability and practical use. To overcome this, we present a methodology to support the easy development and understanding of complex Statecharts. Central to our approach is the definition of a Statechart Normal Form (SNF), which provides a standardized layout that is compact and makes systematic use of secondary notations to aid readability. This concept is extended to dynamic Statecharts.

1. Introduction

Statecharts [4] provide an effective graphical notation, not only for the specification and design of reactive systems, but also for the simulation of the modeled system behavior. However in realistic systems, one is confronted with large and unmanageable graphics due to a high number of components or from interaction and interdependencies. A problem is that existing modeling tools do not offer good mechanisms for abstracting or condensing Statechart representations. The problem becomes even more dramatic when starting to simulate the system, as modular designs typically instantiate Statecharts several times, and each instance may have its own simulation status.

2. A Statechart Normal Form (SNF)

When coding in textual programming languages, it is common practice to structure the text according to some formatting conventions, for example regarding indentation. However, as has already been observed for example by Petre [6], the use of secondary notations in graphical modeling is still underdeveloped. Regarding Statecharts, existing style guidestend to focus on rather basic aspects such as the maximal number of states per chart.

Based on aesthetic criteria [2] and on the meaning underlying the graphical Statechart elements, we have devised rules for the layout of these elements, thus enforcing a standardized use of secondary notations. These rules effectively define a SNF. As an example, consider the Statechart for a simple traffic light controller presented in Figures 1a and 1b. The SNF variant is more compact and lets the viewer focus on the functionality of the Statechart.

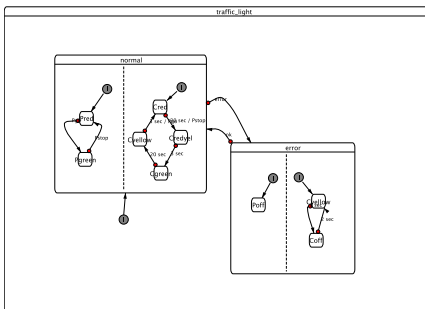
3. Dynamic Statecharts

Statechart modeling tools generally support Statechart *simulation*, where the system under development (SUD) is subjected to some input stimuli, and the Statechart model is animated according to the current configuration of the SUD. The paradigm generally offered is that the Statechart is shown as drawn by the user, and active states and transitions are marked in a specific color. This is fine if the model is small enough to be visible on the screen in its entirety; it becomes problematic for realistic, larger models.

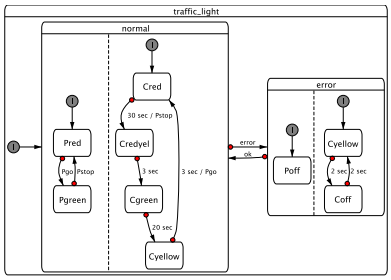
We here present an alternative paradigm for visualizing Statecharts during simulation. The basic idea is to dynamically construct a view of the system model that includes all active states (the *focus*) and their parent states (the *context*); all other states are hidden. This constitutes a dynamic variant of the semantic focus-and-context representation [5]. Compared to the SNF introduced in Section 2, this is a normal form that not only considers the static structure of a Statechart, but also a specific configuration that the system is in. We call this a *Dynamic Statechart Normal Form* (DSNF), which lead to *dynamic Statecharts*. An example sequence of dynamic Statecharts is shown in the Figures 1c, 1d, and 1e.

4. The *KIEL* Modeling Environment

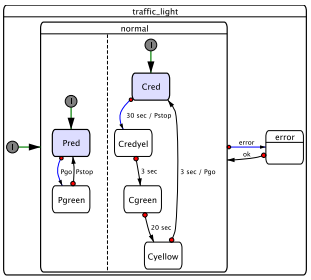
The *Kiel Integrated Environment for Layout* (*KIEL*) employs a generic concept of Statecharts, which can be



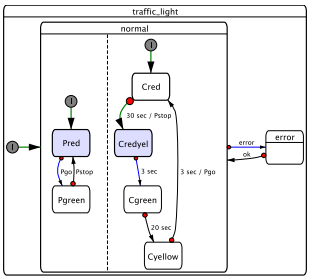
(a) Original, manual layout (imported from Esterel Studio).



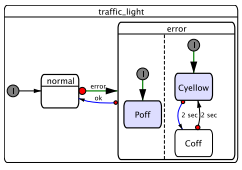
(b) Layout conforming to the SNF.



(c) Entering the initial state—all lights red.



(d) Cars get a red/yellow light.



(e) Entering the error state.

Figure 1: A traffic light example, illustrating the Statechart Normal Form (SNF) and the dynamic simulation (DSNF).

adapted to specific notations and semantics, and it can read in Statecharts that were created using other modeling tools. The currently supported dialects are those from Esterel Studio and from Simulink/Stateflow. *KIEL* also provides an editor to create Statecharts from scratch or to modify imported Statecharts, and it provides a simulator.

Automated Layout of Statecharts: We chose to adopt the layout framework GraphViz [3], which is a collection of tools implementing several graph layout algorithms. Efficiency was a high priority as even for large models the on-the-fly layout of dynamic Statecharts should not noticeably slow down simulations.

Simulation in *KIEL*: *KIEL* can simulate Statecharts, according to the semantics of the Safe State Machines (SSMs) used in Esterel Studio or alternatively according the semantics of Stateflow. In addition to the usual animation of static model views, with highlighting active states and transitions, *KIEL* also provides the dynamic Statechart mechanism outlined in Section 3.

5. Future work

Preliminary feedback regarding the concept of SNFs and dynamic Statecharts has been quite positive. However, we would like to perform more systematic studies on this, also employing expertise from cognitive psychology. We are experimenting with a transformation mechanism that converts Esterel [1] programs into equivalent SSMs; we also work on an alternative textual Statechart description language that is inspired by the dot format employed by GraphViz. This then might result in the best of the textual and the graphical worlds—the efficiency and maintainability of textual entry and the clarity and beauty of visual display.

References

[1] G. Berry and G. Gonthier. The Esterel Synchronous Programming Language: Design, Semantics, Implementation. *Science of Computer Programming*, 19(2):87–152, 1992.

[2] R. Castelló, R. Mili, and I. G. Tollis. A Framework for the Static and Interactive Visualization for Statecharts. *Journal of Graph Algorithms and Applications*, 6(3):313–351, 2002.

[3] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software—Practice and Experience*, 30(11):1203–1234, 2000.

[4] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[5] O. Köth and M. Minas. Structure, Abstraction and Direct Manipulation in Diagram Editors. In M. H. et. al., editor, *LNAI 2317*. Springer Verlag, 2001.

[6] M. Petre. Why looking isn't always seeing: readership skills and graphical programming. *Communications of the ACM*, 38(6):33–44, June 1995.