Sequential Constructiveness and
SCCharts for Safety-Critical Applications*

Reinhard von Hanxleden, Björn Duderstadt, Insa Fuhrmann, Christian Motika, Steven Smyth
Dept. of Computer Science, Kiel University, 24098 Kiel, Germany
{rvh, bdu, ima, cmot, ssm} @informatik.uni-kiel.de

Michael Mendler, Joaquín Aguado
Dept. of Computer Science, Bamberg University, 96045 Bamberg, Germany
{michael.mendler, joaquin.aguado} @uni-bamberg.de

Stephen Loftus-Mercer, Owen O’Brien
National Instruments, Austin, TX 78759, USA
{stephen.loftus-mercer, owen.o’Brien} @ni.com

Abstract: We present a new visual language, SCCharts, designed for the model-driven engineering of safety-critical reactive systems. SCCharts provide determinate concurrency based on a synchronous model of computation (MoC), without restrictions common to previous synchronous MoCs.

SCCharts [vHDM+14] use a statechart notation [Har87] and provide determinate concurrency, based on a synchronous model of computation (MoC) [And04]. We lift earlier limitations on sequential accesses to shared variables, by leveraging the sequentially constructive (SC) MoC [vHMA+14, AMvHF14]. In essence, the SC MoC extends the classical synchronous MoC by allowing variables to be read and written arbitrarily as long as sequentiality expressed in the program provides sufficient scheduling information to exclude race conditions. Previous approaches have excludes test-then-set code such as “if (!initialized) { init(); initialized = true; }” because it requires a write after a read. The SC MoC permits a compiler to safely admit such statements.

The semantics and key features of SCCharts are defined by a very small set of language elements, the Core SCCharts, consisting of state machines plus fork/join concurrency. Conversely, Extended SCCharts contain a rich set of advanced features, such as different abort types, signals, history transitions, etc., as illustrated in Fig. 1; all of these can be reduced via model-to-model transformations into Core SCCharts [MSvH14]. This approach enables a simple yet efficient compilation strategy and aids verification and certification. An open-source, class-room tested SCChart editor/compiler is available in the Eclipse-based modeling framework KIELER1.

*This work was supported by the German Science Foundation (DFG HA 4407/6-1 and ME 1427/6-1) as part of the PRETSY project.

1http://www.informatik.uni-kiel.de/rtsys/kieler/
Figure 1: Syntax overview of SCCharts (from \[vHDM+14\]). The upper region uses Core SCCharts language elements only, the lower region illustrates Extended SCCharts features.

References


