SC2SCFL: Automated SystemC to \(\text{SystemC}^{\text{FL}}\) Translation

Ka Lok Man\(^1\), Andrea Fedeli\(^2\), Michele Mercaldi\(^3\), Menouer Boubekeur\(^1\), and Michel Schellekens\(^1\)

\(^1\) Centre for Efficiency-Oriented Languages (CEOL), Department of Computer Science, University College Cork (UCC), Cork, Ireland
SystemCFL@gmail.com, m.boubekeur@cs.ucc.ie, m.schellekens@cs.ucc.ie
\(^2\) STMicroelectronics, Agrate (Milan), Italy
andrea.fedeli@st.com
\(^3\) M.O.S.T., Turin, Italy
michele.mercaldi@most.it

Abstract. \(\text{SystemC}^{\text{FL}}\) is the formalisation of a reasonable subset of SystemC based on classical process algebras. During the last few years, \(\text{SystemC}^{\text{FL}}\) has been successfully used to give formal specifications of SystemC designs. For formal analysis purposes, so far, users have been required to transform manually their SystemC codes into corresponding \(\text{SystemC}^{\text{FL}}\) specifications. To verify some desired properties of \(\text{SystemC}^{\text{FL}}\) specifications using existing formal verification tools (e.g. NuSMV and SPIN), similarly, manual translations have been needed for turning \(\text{SystemC}^{\text{FL}}\) specifications into corresponding terms of the input language (e.g. SMV and PROMELA) of the selected formal verification tool. Since manual transformation and translations between SystemC codes, \(\text{SystemC}^{\text{FL}}\) specifications, and various formalisms are quite laborious and therefore error-prone, these translations have to be made as much automatic as possible. The first step of the research in these directions is to automate the transformation from SystemC codes to \(\text{SystemC}^{\text{FL}}\) specifications. In this paper, we present SC2SCFL (an automatic translation tool), which converts SystemC codes into corresponding \(\text{SystemC}^{\text{FL}}\) specifications.

1 Introduction

SystemC \(^1\) is a modelling language consisting of C++ class library and a simulation kernel for HDL designs, encompassing system-level behavioural descriptions down to Register Transfer Level (RTL) representations. Nowadays, SystemC is becoming the de-facto standard for system level modelling and design. Although SystemC has been successfully applied in many different industrial areas, and some attempts to apply formal methods to verify SystemC descriptions have been made, it still does miss the possibility of formal reasoning of descriptions.

\(\text{SystemC}^{\text{FL}}\) (SCFL in ASCII format) \(^2\) is the formalisation of a reasonable subset of SystemC based on the classical process algebras Algebra of Communicating Processes (ACP) \(^5\) and A Timed Process Algebra for Specifying Real-Time Systems (ATP) \(^6\). The semantics of \(\text{SystemC}^{\text{FL}}\) has been defined by means of deduction rules in a
Structured Operational Semantics (SOS) [7] style that associates a time transition system (TTS) with a SystemCFL process. The introduction of SystemCFL since three years ago initiated an attempt to extend the knowledge and experience collected in the field of process algebras to system-level modelling and design.

SystemCFL is aimed at giving formal specifications of SystemC designs and to perform formal analysis of SystemC processes. Furthermore, SystemCFL is a single formalism that can be used for specifying concurrent systems, finite state systems, and real-time systems (as in SystemC). Desired properties of these systems specified in SystemCFL can be verified with existing formal verification tools by translating them into different formalisms that are the input languages of formal verification tools. Hence, SystemCFL can be purportedly used for formal verification of SystemC designs. For instance, safety properties of concurrent systems specified in SystemCFL can be verified (see [8]) by translating those systems to PROMELA [9], which is the input language of the SPIN Model Checker [9]. Similarly, [10] reported that some desired properties of finite state systems specified in SystemCFL can be fed into the SMV Model Checker [11] to verify them. Furthermore, a formal translation was defined in [12] from SystemCFL to a variant (with very general settings) of timed automata [13]. The practical benefit of the formal translation from a SystemCFL specification (describing real-time systems) to a timed automaton is to enable verification of timing properties of the SystemCFL specifications using existing verification tools for timed automata, such as UPPAAL [14].

During the last few years, SystemCFL has been successfully used to give formal specifications of SystemC designs (see also [3, 4, 15]). For formal analysis purposes, so far, users have been required to manually transform their SystemC codes into corresponding SystemCFL specifications. To verify some desired properties of SystemCFL specifications using existing formal verification tools (see also [8, 10, 12]), similarly, manual translations have been needed for turning SystemCFL specifications into corresponding terms of the input language of the selected formal verification tool. Since manual transformation and translations between SystemC codes, SystemCFL specifications, and various formalisms are quite laborious and therefore error-prone, these translations have to be automated.

For the sake of simplicity, our first goal (of the research in these directions as already reported in [16]) is to develop an automatic translation tool which converts SystemC codes (mainly untimed) into corresponding SystemCFL specifications that can be further mapped to the input languages of several formal verification tools (e.g. SPIN and NuSMV). SC2SCFL is such an automatic translation tool.

Over the years, automatic translation tools from SystemC to other description languages have been developed (e.g. [17]). To our knowledge, this is the first article to report an automatic translation tool from SystemC to a formal language (i.e. SystemCFL).

This paper is organised as follows. In Section 2 we give a brief overview of a subset of SystemCFL formalism that is relevant for the use in this paper. Section 3 captures the main ideas of our proposed translation from SystemC to SystemCFL. Section 4 describes the architectures of the automatic translation tool SC2SCFL. A translation example of using SC2SCFL is given in Section 5. We also illustrate our practical interest of verification of SystemCFL specifications, in Section 6, by translating the SystemCFL specification shown in Section 5 to the equivalent NuSMV [18] specification which is