A formal semantics of extended hierarchical state transition matrices using CSP#

Yoriyuki Yamagata¹, Weiqiang Kong², Akira Fukuda², Nguyen Van Tang¹,
Hitoshi Ohsaki¹, and Kenji Taguchi¹

¹ National Institute of Advanced Industrial Science and Technology (AIST), Nakoji 3-11-46, Amagasaki, Hyogo 661-0974, Japan
² Kyushu University, Fukuoka, Japan

Abstract. The extended hierarchical state transition matrices (EHSTMs) are a table-based modelling language frequently used in industry for specifying behaviours of systems. However, assuring correctness, i.e., having a design satisfy certain desired properties, is a non-trivial task. To address this problem, a model checker dedicated to EHSTMs called Garakabu² has been developed. However, there is no formal justification for Garakabu², since its semantics has never been fully formalised. In this paper, we give a formal semantics to EHSTMs by translating them into CSP, Communicating Sequential Processes. Among the variants of CSP, we use CSP#, which is the modelling language used by PAT model checker, as a target of translation. Our semantics covers most of the features supported by Garakabu². We manually translate the small examples of EHSTMs to CSP#, and verify them by PAT. We also verify the examples directly using Garakabu² and show that the results are same. The experiments also indicate that verification using our translation and PAT is much faster than that of Garakabu² in some cases.

Keywords: Embedded systems, Software modelling, Formal semantics, Model checking, CSP

1. Introduction

The extended hierarchical state transition matrices (EHSTMs) [Wat98] are a table-based modelling language for specifying behaviours of systems. In an EHSTM, the horizontal axis declares the possible states of a system under consideration; the vertical axis declares the possible events that may occur to the system, and a row-column intersection cell declares behaviours of the system when the designated event is dispatched (occurs) in the designated state. Further, EHSTMs can form a hierarchy. At a point in its execution, the parent (upper) EHSTM $T_1$ can “delegate” all invocations of events to a child (lower) EHSTM $T_2$ and wait until $T_2$ “returns” the control. $T_2$ in turn, waits until $T_1$ calls $T_2$, then processes all events which occur until it reaches the “return” cell. In addition, events and states can also form hierarchies. A whole system can be described by defining each of its sub-systems as a hierarchy of EHSTMs (called a task) and by the tasks that communicate via shared variables or message passing.

Correspondence and offprint requests to: Y. Yamagata, E-mail: yoriyuki.yamagata@aist.go.jp
EHSTMs has been used frequently in industry to develop software designs. According to a survey [Ass12] which Japan Embedded System Technology Association (JASA) conducted at Embedded Technology 2010 (ET 2010), one of the largest conventions for embedded technology in Japan, 45% of developers are using state transition tables as a design method. In addition, the modelling tool ZIPC, which employs EHSTMs to describe software designs, has top market share among non-UML based modelling tools. Despite this popularity, assuring correctness, i.e., having the EHSTM design satisfy certain desired properties remains a non-trivial task. On the one hand, designers may inadvertently introduce subtle logical errors into an EHSTM design, especially when the design involves multiple communicating tasks, which makes it difficult for designers to maintain an overall image of the whole design. On the other hand, there is a lack of mechanised verification support for EHSTM designs, e.g., ZIPC provides facilities for syntactic check only.

To provide a mechanised verification support, Garakabu2 [KSK+11, KKQ+11] has been developed. Garakabu2 translates the EHSTM design into formulae whose validity can be checked by an SMT solver. Thus, Garakabu2 performs a bounded model checking [BCCZ99] of EHSTM designs by using a Satisfiability Modulo Theories (SMT) solving technique [BSST]. More specifically, in this approach, all execution sequences within a given bound of an EHSTM design and the negation of a LTL-property to be checked, are encoded into a quantifier-free formula whose satisfiability is to be checked with respect to some decidable background theories, such as the theory of integers and the theories of various data structures such as arrays. Satisfiability of the resulting formula can be determined by state-of-the-art SMT solvers. If satisfied, a model (interpretation to all the variables involved) of the formula is a witness to some bad behaviours of the design.

One of the remaining problems is, then, to justify the encoding from EHSTM designs to SMT formulae. There are three reasons why justification of the encoding is difficult. The first reason is that the semantics of EHSTMs is given by a simulator and generated C codes. Thus, there are ambiguities in the semantics of EHSTMs. The second is that EHSTMs introduce non-trivial extension such as hierarchies of tables and events to a simple transition table. Therefore, we cannot just look to the translation and believe its correctness. The third reason is that the rules for encoding EHSTMs to SMT formulae are buried in C++ codes and have never been fully formalised (partial formulation, however, can be found in [KSK+11, KKQ+11]).

To address this problem, we introduce a translation of a subset of EHSTMs into CSP# [SLD09] which is based on CSP [Hoa04, RHB98]. Among other formal methodologies, we choose CSP since CSP is a language that models concurrent processes, and thus is suitable for interpreting EHSTM designs which are inherently parallel. Further, CSP has a long history of research, and is equipped with a well-defined operational and denotational semantics. Thus, the translation gives EHSTMs a formal semantics, albeit in an indirect way. CSP is a concise and relatively high-level language; thus, CSP can facilitate discussions among non-experts. Finally, CSP is supported by good verification tools such as FDR2 (http://www.fsel.com) and PAT [SLD09, LSD11]. The reason that we choose CSP# is that it supports global variables and message queues directly, therefore CSP# makes our translation simpler.

We translate only a subset of EHSTMs; this subset covers the basic functionality supported by Garakabu2. By translating EHSTMs into CSP#, we determine the semantics of composed statements, event virtual frames, event hierarchies, state virtual frames and parallel states with/without synchronisation. Also, we consider the problem of the atomicity of compound actions. Further, we compare the results and time required for verification of EHSTMs by Garakabu2, and our translation of these EHSTMs by PAT. We find that by our approach, verification is much faster than Garakabu2, at least on the small examples used in our experiments.

As stated earlier, Garakabu2 is an automated model checker for EHSTMs. Garakabu2 translates EHSTM designs into formulae which are interpreted by SMT solvers, and performs bounded model checking. The translation from EHSTMs into a formal language by Garakabu2 can be seen as a sort of formal semantics of EHSTMs. However, translation rules are buried in C++ codes and have never been fully formalised. Although Garakabu2 handles such as event virtual frames, event hierarchy and state virtual frame, the method to handle these features has not been published to date. By using CSP# we have an advantage in that we can use all CSP-related research for EHSTMs. In particular, we apply PAT to our translation, and find that the time required for verification is greatly reduced from Garakabu2 in some cases.

1.1. Related work

The primary motivation and contribution of our work is to define a formal semantics to EHSTMs, which is, however, not discussed satisfactorily in early works on Garakabu2 [KKQ+11, KSK+11, KLY+12]. The semantics is given in our work through a translation from EHSTMs into CSP#. As a by-product or another