A Semantics-Based Approach for Mapping Specification Notations to Analysis Tools

Jianwei Niu
School of Computer Science
University of Waterloo

Research Areas Requirement engineering, Formal methods, Computer-aided verification, Model-based notations

1 Problem Statement and Related Work

Model-based notations (e.g. statecharts [7, 8], RSML [14], CSP [10], CCS [17] and LOTOS [11]) have been widely used by practitioners to describe software systems. They have intuitive step semantics and their composition operators provide facilities for decomposing large problems into modules and for expressing concurrency, synchronization and communications among modules. Many well-established analysis tools, such as model checkers, have been successfully applied to verify automatically model-based specifications. In order to analyze specifications in a model-based notation, either an analyzer is constructed specifically for the notation or a translator is written from the notation to the input language of an existing analysis tool. Translation is a common approach for reusing existing tools to analyze a specification without rewriting it by hand [1, 2]. However, a translator needs to be constructed for each notation and needs to be changed whenever the syntax or the semantics of the notation evolves.

To help tackle these problems, we and other researchers propose approaches to generate analyzers automatically from a description of a notation's semantics [4, 6, 20]. In these approaches, the semantics of a notation are defined and fed into an automatic analyzer generator and a compiler for that notation is produced. The compiler takes a specification in that notation as an input, and it automatically generates a transition-relation for the specification. Transition-relations form the basis of most analyzers, so a transition-relation representation of the specification can be checked using many automatic analysis tools, such as model checkers.

Problem Statement: Because the semantics of model-based notations are complex,

formalizing them in a semantic description language such as structural operational semantics, higher-order logic, or hyper-graph rules is challenging, error-prone and a barrier to the utility of current approaches for generating analyzers. This work aims at simplifying the expression of notations' semantics.

2 Research Objective

In order to ease the effort required to define semantics, we propose a semantics template that captures the common behaviour of different notations and parameterizes notations' distinct behaviours [19]. The semantics of a particular notation are expressed as an instantiation of this template that specializes a notation's choice of both its step semantics and composition semantics. The main motivation of this work is to ease the mapping of new notations or notation variants to analysis tools.

Thesis Statement: We can generate model compilers automatically from descriptions of notations' semantics using our semantics template. Template-based semantics are succinct definitions that are sufficiently expressive to capture the semantics of many model-based notations. Such an approach can effectively generate a model compiler for a notation, which in turn generates a transition-relation for a specification. The transition-relation then can be used as an input to formal analysis tools.

The rest of this article is organized as follows. Section 3 describes the current research result. Section 4 discusses the methods that will be used in developing the semantics-based approach for instantiating a model compiler. Section 5 proposes the validation study of this work.

3 Research Progress

After surveying a set of popular model-based notations, CSP [10], CCS [17], LOTOS [11], basic transition systems [16], SDL [13], ESTELLE [12], and several variants of statecharts [7, 8, 14], we captured the essential aspects of each notation as attributes of a common, composable, hierarchical transition system (HTS). An HTS is a non-concurrent, hierarchical, extended finite state machine, which is adapted from basic transition systems and statecharts. We use an HTS to model the basic component of a model-based system. The step semantics of an HTS is expressed as a relation between consecutive snapshots. A snapshot is an execution point of the system, which contains four current elements (states, internal and external events, and variable values) and four auxiliary elements (one for each of the current elements). The auxiliary elements accumulate data about states, internal and external events, and variable values to accommodate the differences in step semantics among notations. We have formally defined an operational semantics template for model-based no-

tations, in which the step semantics of an HTS are parameterized by parameter functions for the snapshot elements [19]. An instantiation of this template represents a notation's choice of which transitions are enabled and which transition executes in a given state. For example, one parameter function computes the set of enabling states of an HTS using the snapshot elements, current states and auxiliary states, and another parameter function computes the set of enabling external events using current external events and auxiliary external events. The template identifies a collection of 12 parameter functions that the user must define.

We also identified seven well-used composition operators: interleaving, parallel, environmental and rendezvous synchronizations, sequence, choice, and interrupt. A collection of HTSs can be composed into a specification using these composition operators, which express concurrency and communication among HTSs. We have defined the semantics of these composition operators separately, as relations that constrain how components can execute in parallel, transfer control to one another, and exchange events and data; the definitions of these composition operators use the template parameters to preserve in composition the notation-specific behaviour. Other model-based notations may require different composition operators, our semantics template provides a pattern for defining additional composition operators in terms of how the elements of the snapshot are affected by the operators.

The semantics template for model-based notations forms the theoretical foundation of my thesis. Most of the semantics of the notations that we initially surveyed can be expressed as instantiated variants of our template. A side effect of our work is a relatively clean separation of a notation's execution semantics from its composition operators, thereby simplifying the definitions of both step semantics and composition. Another effect of this work is progress towards standardizing the semantics of model-based notations: by parameterizing the notations' distinct behaviour, it is easier to see precisely the difference in the semantics between two languages.

4 Methods

I propose to use this theoretical template to generate automatically a model compiler for model-based notations. The formal definitions of the step semantics template and composition operators for model-based notations provide the theoretical foundations for this work. The next step is to implement the approach using higher-order logic. I am going to embed the formal definitions of our semantics template and composition operators into S+, a higher-order logic language supported by Fusion [3]. Fusion is a verification tool suite for analyzing a specification written in the notation whose semantics are represented in the S+ language.

The semantics of a particular model-based notation can be defined by instantiating the parameter functions of our template and either mapping the notation's composition operators

to our pre-defined composition operators or defining new composition operators using our operators as a guide. Our approach will take as input the semantics template embedded in S+ and the parameter functions of a specific notation, which instantiate the template. We use symbolic functional evaluation (SFE) [5] to expand the meaning of a specification as defined by the notation's semantics, producing as output the specification's transition relation. Various model checkers can then be used to analyze the generated transition-relation.

5 Validation

After implementing the proposed semantics-based approach, I plan to evaluate its effectiveness, succinctness and expressiveness. The production cell problem [15] is a good candidate for case study. I will evaluate the effectiveness of our approach by analyzing specifications of the production cell in various model-based notations using analysis tools from the semantics of the notations. I hope to show that our approach is succinct and easy to use in that a model-based notation's semantics can be represented by instantiating our step semantics template and mapping from the notation's composition operators to our well-defined composition operators. I intend to demonstrate the utility of the approach by attempting to generate analyzers for notations from our original set of languages, such as statecharts and LOTOS. If the essential features of these notations can be represented by our approach and model compilers can be generated automatically, our approach is successful. I also plan to demonstrate the expressiveness of the approach by describing the semantics for other model-based notations that were not included in our initial survey list, such as Petri-Net [18] and SCR [9]. I hope to show that our approach can accommodate these notations, whose semantics are quite different from our initial notations, with little or no changes to our template's set of snapshot elements and parameter functions.

6 Acknowledgements

I deeply thank my thesis committee (Jo Atlee, Dan Berry, Nancy Day, Andrew Malton, and John Thistle of the University of Waterloo) for helpful supervision, suggestion and discussion.

References

[1] J. M. Atlee and J. Gannon. State-based model checking of event-driven system requirements. *IEEE Trans. on Soft. Eng.*, 19(1):24–40, January 1993.

[2] W. Chan, R. J. Anderson, P. Beame, S. Burns, F. Modugno, and D. Notkin. Model checking large software specifications. *IEEE Trans. on Soft. Eng.*, 24(7):498–519, 1998.

[3] N. Day. Fusion User's Guide. Unpublished.

[4] N. Day. *A Framework for Multi-Notation, Model-Oriented Requirements Analysis*. PhD thesis, University of British Columbia, October 1998.

[5] N. A. Day and J. J. Joyce. Symbolic functional evaluation. In *Proc. of 12th Intl. Conf. on Theorem Proving in Higher Order Logics (TPHOL)*, number 1690 in LNCS, pages 341–358. Springer, 1999.

[6] L. K. Dillon and K. Stirewalt. Lightweight analysis of operational specifications using inference graphs. In *ICSE*, pages 57–67. IEEE Comp. Soc, 2001.

[7] D. Harel et al. On the formal semantics of statecharts. In *Symp. on Logic in Comp. Sci.*, pages 54–64, 1987.

[8] D. Harel and A. Naamad. The Statemate semantics of statecharts. *ACM Trans. on Soft. Eng. Meth.*, 5(4):293–333, 1996.

[9] K. L. Heninger, D. Parnas, J. E. Shore, and J. W. Kallander. Software requirements for the A-7E aircraft. Technical Report 3876, Naval Research Lab, 1978.

[10] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, UK, 1985.

[11] ISO8807. LOTOS - a formal description technique based on the temporal ordering of observational behaviour. Technical report, ISO, 1988.

[12] ISO9074. ESTELLE - a formal description technique based on an extended state transition model. Technical report, ISO, 1989.

[13] ITU-T. Recommendation Z.100. Specification and Description Language (SDL). Technical Report Technical Report Z-100, International Telecommunication Union - Standardization Sector, 1999.

[14] N. G. Leveson, M. P. E. Heimdahl, H. Hildreth, and J. D. Reese. Requirements specification for process-control systems. *IEEE Trans. on Soft. Eng.*, 20(9), September 1994.

[15] C. Lewerentz and T. Lindner, editors. *Case study "Production Cell": A Comparative Study in Formal Software Development*. Forschungszentrum Informatik, 1994.

[16] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1991.

[17] R. Milner. *Communication and Concurrency*. Prentice Hall, New York, 1989.

[18] T. Murata. Petri Nets: Properties, analysis and applications. *Proc. of the IEEE*, 77(4):541–580, April 1989.

[19] J. Niu, J. M. Atlee, and N. A. Day. Composable semantics for model-based notations. In *Proc. of the 10th ACM SIGSOFT Symp. on Foundations of Soft. Eng. (FSE)*, pages 149–158, 2002.

[20] M. Pezzè and M. Young. Constructing multi-formalism state-space analysis tools. In *ICSE*, pages 239–249. ACM Press, 1997.