Synthesizing Modal Transition Systems from Triggered Scenarios

German Emir Sibay, Victor Braberman, Sebastian Uchitel, *Member, IEEE Computer Society*, and Jeff Kramer, *Member, IEEE Computer Society*

Abstract—Synthesis of operational behavior models from scenario-based specifications has been extensively studied. The focus has been mainly on either existential or universal interpretations. One noteworthy exception is Live Sequence Charts (LSCs), which provides expressive constructs for conditional universal scenarios and some limited support for nonconditional existential scenarios. In this paper, we propose a scenario-based language that supports both existential and universal interpretations for conditional scenarios. Existing model synthesis techniques use traditional two-valued behavior models, such as Labeled Transition Systems. These are not sufficiently expressive to accommodate specification languages with both existential and universal scenarios. We therefore shift the target of synthesis to Modal Transition Systems (MTS), an extension of labeled Transition Systems that can distinguish between required, unknown, and proscribed behavior to capture the semantics of existential and universal scenarios. Modal Transition Systems support elaboration of behavior models through refinement, which complements an incremental elicitation process suitable for specifying behavior with scenario-based notations. The synthesis algorithm that we define constructs a Modal Transition System that uses refinement to characterize all the Labeled Transition Systems models that satisfy a mixed, conditional existential and universal scenario-based specification. We show how this combination of scenario language, synthesis, and Modal Transition Systems supports behavior model elaboration.

Index Terms—Scenarios, MTS, synthesis, partial behavior models

1 Introduction

Operational behavioral models such as labeled Transition Systems (LTSs) are convenient formalisms for modeling and reasoning about system behavior at the architectural level. These models provide a basis for a wide range of automated (and semi-automatic) analysis techniques, such as model-checking, simulation, and animation.

One of the limitations of operational behavior modeling is the complexity of building the models in the first place. Operational behavioral model construction remains a difficult, labor-intensive task that requires considerable expertise. To address this, a wide range of techniques for supporting (semi-)automated synthesis of operational behavior models has been investigated. In particular, synthesis from scenarios and use cases has been studied extensively [1], [2], [3], [4], [5].

Scenario-based specifications such as Message Sequence Charts (MSCs) [6] describe how system components, the environment, and users interact in order to provide system level functionality. Their simplicity and intuitive graphical representation facilitate stakeholder involvement, making them popular for requirements elicitation. Model synthesis from scenario-based specifications facilitates early analysis, validation, and incremental elaboration of behavior models.

A range of scenario description languages and associated behavior model synthesis algorithms have been developed (e.g., [1], [7], [8]). Although they differ in many aspects, a noteworthy semantic distinction is whether scenarios are interpreted as existential or universal statements. An existential scenario provides an example of system behavior, one that the system-to-be is required to provide. A universal scenario provides a rule that all system behavior is expected to satisfy. Although each approach is typically geared to one interpretation or the other, some languages, notably Live Sequence Charts (LSCs) [3], provide syntactic and semantic support for both interpretations. The motivation is that during the requirements process, there is a progressive shift from existential statements in the form of examples and use cases to universal statements in the form of declarative properties. A scenario-based language that supports both interpretations is better equipped to support this shift.

1.1 Existential Triggered Scenarios (eTSs)

Despite the variety of existing approaches, no language and associated synthesis algorithm is suitable for describing conditional existential scenarios. Consider the statement "if the user inserts a valid card into the ATM, and then enters the correct password, she/he shall be able to request cash and have it dispensed by the ATM." This statement is existential in that it provides an example of system execution. It is also conditional in the sense that requesting

- *G.E. Sibay and J. Kramer are with the Department of Computing, Imperial College London, Room 504, 180 Queen's Gate, Huxley Building, SW7 2B7 London, United Kingdom. E-mail: gsibay@doc.ic.ac.uk, j.kramer@imperial.ac.uk.*
- *V. Braberman is with the Department of Computing, FCEN, University of Buenos Aires, Pabellón 1, Ciudad Universitaria, Intendente Güiraldes 2160, Buenos Aires (C1428EGA), Argentina. E-mail: vbraber@dc.uba.ar.*
- *S. Uchitel is with the Department of Computing, Imperial College London, London SW7 2AZ, United Kingdom, and with the Department of Computing, FCEN, University of Buenos Aires, Pabellón 1, Ciudad Universitaria, Intendente Güiraldes 2160, Buenos Aires (C1428EGA), Argentina. E-mail: suchitel@dc.uba.ar.*

Manuscript received 11 Mar. 2010; revised 7 Mar. 2011; accepted 13 Sept. 2012; published online 21 Sept. 2012.
Recommended for acceptance by M. Kwiatkowska.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2010-03-0070.
Digital Object Identifier no. 10.1109/TSE.2012.62.

7 Conclusion

In this paper, we have defined a scenario-specification language which includes support for describing triggered existential and universal scenarios. We have also defined a synthesis algorithm that constructs MTS models which characterize via refinement all LTS models that conform both to the existential and universal aspects of the scenario-based description.

A novel aspect of the approach is the use of triggered existential scenarios which have a branching semantics. This is in line with existing informal scenario-based and use case-based approaches to requirements engineering exploiting the expressive power of MTS in an operational behavior model.

The approach supports behavior elaboration through the analysis and refinement of underspecified system behavior using MTS merging, model checking, inspection, and animation, moving from examples to comprehensive descriptions during the behavior elaboration process.

In future work, we intend to continue to develop and integrate support for elicitation and elaboration of behavior models using MTS. In particular, we are investigating the use of learning, in the form of Inductive Logic Programming [40], to aid the elaboration process. We aim to develop techniques and tools to support identifying, providing feedback, and resolving inconsistencies in the process of merging MTS that result from scenario-based specifications.

Acknowledgments

The work reported herein was partially supported by ERC StG PBM-FIMBSE, CONICET PIP112-200801-00955KA4, UBACYT W0813, PICT 1774, PICT PAE 2272, PICT PAE 2278, and MEALS 295261. Finally, the authors would like to thank the anonymous reviewers for their careful and insightful reviews that greatly improved the quality of this paper.

References

[1] I. Kruger, "Distributed System Design with Message Sequence Charts," PhD dissertation, Technical Univ. of Munich, 2000.
[2] S. Uchitel, J. Kramer, and J. Magee, "Incremental Elaboration of Scenario-Based Specifications and Behaviour Models Using Implied Scenarios," *ACM Trans. Software Eng. and Methodology*, vol. 13, no. 1, pp. 37-85, 2004.
[3] D. Harel and R. Marelly, *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer, 2003.
[4] Y. Bontemps, P. Heymans, and P.-Y. Schobbens, "From Live Sequence Charts to State Machines and Back: A Guided Tour," *IEEE Trans. Software Eng.*, vol. 31, no. 12, pp. 999-1014, Dec. 2005.
[5] T. Ziadi, L. Helouet, and J.-M. Jezequel, "Revisiting Statechart Synthesis with an Algebraic Approach," *Proc. 26th IEEE Int'l Conf. Software Eng.*, pp. 242-251, 2004.
[6] ITU, "Recommendation z.120: Message Sequence Charts," 2000.
[7] B. Sengupta and R. Cleaveland, "Triggered Message Sequence Charts," *IEEE Trans. Software Eng.*, vol. 32, no. 8, pp. 587-607, Aug. 2006.
[8] W. Damm and D. Harel, "LSCs: Breathing Life into Message Sequence Charts," *Proc. Third Int'l Conf. Formal Methods for Open ObjectBased Distributed Systems*, vol. 139, 1999.
[9] K. Zachos, N. Maiden, and A. Tosar, "Rich-Media Scenarios for Discovering Requirements," *IEEE Software*, vol. 22, no. 5, pp. 89-97, Sept./Oct. 2005.
[10] R.M. Keller, "Formal Verification of Parallel Programs," *Comm. ACM*, vol. 19, pp. 371-384, July 1976.
[11] R. Milner, *Communication and Concurrency*. Prentice-Hall, 1989.
[12] K. Larsen and B. Thomsen, "A Modal Process Logic," *Proc. Third Ann. IEEE Symp. Logic in Computer Science*, pp. 203-210, 1988.
[13] K. Larsen, B. Steffen, and C. Weise, "The Methodology of Modal Constraints," *Formal Systems Specification*, vol. 1169, pp. 405-435, 1996.
[14] M. Huth, R. Jagadeesan, and D.A. Schmidt, "Modal Transition Systems: A Foundation for Three-Valued Program Analysis," *Proc. 10th European Symp. Programming Languages and Systems*, pp. 155-169, 2001.
[15] K.G. Larsen, B. Steffen, and C. Weise, "A Constraint Oriented Proof Methodology Based on Modal Transition Systems," *Proc. First Int'l Workshop Tools and Algorithms for Construction and Analysis of Systems*, pp. 13-28, 1995.
[16] D. Fischbein and S. Uchitel, "On Correct and Complete Merging of Partial Behaviour Models," *Proc. SIGSOFT Conf. Foundations of Software Eng.*, pp. 297-307, 2008.
[17] S. Uchitel, G. Brunet, and M. Chechik, "Synthesis of Partial Behavior Models from Properties and Scenarios," *IEEE Trans. Software Eng.*, vol. 3, no. 35, pp. 384-406, May 2009.
[18] G. Brunet, M. Chechik, D. Fischbein, N. D'Ippolito, and S. Uchitel, "Weak Alphabet Merging of Partial Behaviour Models," *ACM Trans. Software Eng. and Methodology*,
[19] G. Bruns and P. Godefroid, "Generalized Model Checking: Reasoning about Partial State Spaces," *Proc. 11th Int'l Conf. Concurrency Theory*, pp. 168-182, 2000.
[20] D. Fischbein, N. D'Ippolito, G. Sibay, and S. Uchitel, "Modal Transition System Analyser (MTSA)," http://sourceforge.net/projects/mtsa/, 2012.
[21] J. Magee and J. Kramer, *Concurrency—State Models and Java Programs*. John Wiley, 1999.
[22] S. Uchitel and M. Chechik, "Merging Partial Behavioural Models," *Proc. 12th ACM SIGSOFT Int'l Symp. Foundations of Software Eng.*, pp. 43-52, 2004.
[23] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*. MIT Press, 1999.
[24] G. Sibay, S. Uchitel, and V. Braberman, "Existential Live Sequence Charts Revisited," *Proc. 30th Int'l Conf. Software Eng.*, pp. 41-50, 2008.
[25] D. Giannakopoulou and J. Magee, "Fluent Model Checking for Event-Based Systems," *Proc. Ninth European Software Eng. Conf. Held Jointly with 11th ACM SIGSOFT Int'l Symp. Foundations of Software Eng.*, 2003.
[26] R. Alur, K. Etessami, and M. Yannakakis, "Inference of Message Sequence Charts," *IEEE Trans. Software Eng.*, vol. 29, no. 7, pp. 623-633, July 2003.
[27] S. Mauw and M.A. Reniers, "High-Level Message Sequence Charts," *Proc. Int'l Conf. System Design Languages*, pp. 291-306, 1997.
[28] I. Beer, S. Ben-David, C. Eisner, and Y. Rodeh, "Efficient Detection of Vacuity in ACTL Formulas," *Proc. Ninth Int'l Conf. Computer-Aided Verification*, pp. 279-290, 1997.
[29] N. D'Ippolito, D. Fishbein, H. Foster, and S. Uchitel, "MTSA: Eclipse Support for Modal Transition Systems Construction, Analysis and Elaboration," *Proc. OOPSLA Workshop Eclipse Technology Exchange*, pp. 6-10, 2007.
[30] R. van Ommering, F. van der Linden, J. Kramer, and J. Magee, "The Koala Component Model for Consumer Electronics Software," *Computer*, vol. 33, no. 3, pp. 78-85, 2000.
[31] W. Damm and D. Harel, "LSCs: Breathing Life into Message Sequence Charts," *Formal Methods in System Design*, vol. 19, no. 1, pp. 45-80, 2001.
[32] J. Magee, N. Pryce, D. Giannakopoulou, and J. Kramer, "Graphical Animation of Behavior Models," *Proc. 22nd Int'l Conf. Software Eng*, pp. 499-508, 2000.
[33] H. Kugler, M.J. Stern, and E.J.A. Hubbard, "Testing Scenario-Based Models," *Proc. 10th Int'l Conf. Fundamental Approaches to Software Eng.*, pp. 306-320, 2007.
[34] K. Larsen and L. Xinxin, "Equation Solving Using Modal Transition Systems," *Proc. Fifth Ann. IEEE Symp. Logic in Computer Science*, pp. 108-117, 1990.
[35] Y. Bontemps, P.-Y. Schobbens, and C. Löding, "Synthesis of Open Reactive Systems from Scenario-Based Specifications," *Fundamenta Informaticae*, vol. 62, no. 2, pp. 139-169, 2004.
[36] D. Harel and H. Kugler, "Synthesizing State-Based Object Systems from LSC Specifications," *Int'l J. Foundation of Computer Science*, vol. 13, no. 1, pp. 5-51, 2002.