Abstraction in Software Model Checking: Principles and Practice

(Tutorial overview and bibliography)

Dennis Dams

Bell Labs, Lucent Technologies, 600 Mountain Ave, Murray Hill, NJ 07974, USA.
dennis@research.bell-labs.com

Abstract. This paper provides a brief description, including a bibliography, of the SPIN2002 tutorial on abstraction in model checking of software.

1 Introduction

The tutorial assumes familiarity with the principles of model checking ([CGP99]), which is an approach to the formal verification of temporal correctness properties of finite state systems. The starting point of a model checker is a verification model: a formal system description, called *abstract system* henceforth, having a state space that is small enough to render model checking feasible. The goal is to establish correctness of the original system being modelled. When a (more detailed) formal description is also available for this *concrete system*, one can try and formalize the relation between these systems, possibly with the aim of offering automated support for the abstraction process. In the context of model checking, the term *abstraction* refers to methodology, theory, techniques, and tools that deal with the relation between formalized system descriptions at different levels of detail.

Abstraction methodologies are concerned with the *process* of abstraction: Given a concrete system and a property to be checked, how to get to a suitable abstract system? This process typically involves a form of trial-and-error, and depends on rules-of-thumb and ingenuity. Abstraction theory focuses on formalizing the relation between the semantic models of concrete and abstract systems. A prime requirement of such a relation is that it ensures *preservation* of correctness properties: A property checked to be true for the abstract system should also hold for the concrete system being modelled. By abstraction techniques we mean the methods that can be employed to construct abstract systems from concrete ones. These range from slicing and variable hiding to more general, less algorithmic approaches like program transformation based on abstract interpretation, which may require human interaction. There exist several software tools that implement such abstraction techniques. At its front end such a tool offers what is essentially a programming language in which a system description may be entered. The core of the tool consists of a collection of components that implement techniques, sometimes several alternative ones, for abstraction. Also, methodological guidelines may be provided aiding in the selection of a sequence of abstraction steps. At the back end, a verification model is then produced in a form that is accepted by a model checker.

As abstraction is a very broad field, we cannot discuss all relevant approaches. Techniques that can be viewed as instances of abstraction but that will not be further touched upon here include data-independence, (de)compositionality, parameterization, partial order reduction, real time verification, and symmetry techniques. The focus will be mostly on model checking of software source code — as a consequence BBD-based approaches to abstraction will receive less attention.

Much of the tutorial is based on [Dam96].

2 Methodology

There is relatively little research into the methodological aspects of combining model checking and abstraction. Generally, the process follows the cycle that occurs in all approaches to software validation. For the case of model checking the steps are summarized in [CGP99], p. 4: modeling, specification, verification. If the last of these steps fails, then inspection of the counterexample will indicate an error in the system, in the model, or in the specification, leading to a repetition of the steps.

For an approach that combines model checking with formal abstraction, an instance of this cycle is commonly proposed. In this setting, the model can be viewed as the result of applying an abstraction to the concrete system, and thus the triple (system, model, specification) may be replaced by (system, abstraction, specification). A negative answer produced by running a model checker on this may indicate an error in any of the three ingredients. The term *false negative* refers to the case that the abstraction is too coarse — inspection of the counterexample may then suggest a way to refine it.

More or less explicit descriptions of methodologies are found in [BH99, BR01, DHJ$^+$01, Hol01, HS02, LBBO01, WC99], often embedded in reports on case studies, or in descriptions of verification tools by which they are supported. A paper discussing methodological issues in formal methods at a more general level is [Hei98].

3 Theory

Because of its strong roots in the formal methods community, there is a large body of theory on abstraction. Here we focus on papers that provide the common theoretical underpinnings. Papers that provide the foundations for specific techniques and tools may be found through references given in the sections below.

State-transition systems are commonly used as the formal semantics on both the concrete and abstract sides. Results on property-preserving relations between these draw on the theory of formal languages and automata ([HU79]), in particular on results about homomorphisms and language inclusion ([Gin68]), minimization and partition refinement ([BFH$^+$92, GV90, Hop71, KS90, PT87]), and on extensions of automata to infinite words ([Buc60]). The topic of comparative semantics has also been extensively studied in the context of process algebra ([BW90]), see e.g. [DN87, vG90]. In particular the notion of *bisimulation* ([Par81]), weaker equivalences and pre-orders related to it ([GW89, Mil71, Mil80]), and their connection to modal and temporal logic ([ASB$^+$94, BCG88, BFG$^+$91, BR83, Cho95, DNV90, GKP92, GS84, HM80, Kur94, Sti89, vBvES94]) are relevant.

The partition refinement algorithms mentioned above may be used in a *quotient construction* that produces a minimal transition system that is equivalent to the original system under some notion of behavioural (bisimulation-like) equivalence. The starting point for model checking under abstraction is usually a more drastically reduced system which is related to the concrete system through a behavioural pre-order like simulation ([CGL94]). The satisfaction of (temporal) logic formulas over these abstract systems is usually non-standard: properties may evaluate to "unknown" as a result of abstracting away certain information. A similar notion of incomplete information is common in the related area of program analysis and Abstract Interpretation ([CC77, NNH99]). Reasoning with it in terms of modal and temporal logic, in the context of model checking, is a topic that is receiving considerable attention: [BG99, CDE$^+$01, DGG00a, HJS01]. An overview of many-valued modal logics is given in [Fit91, Fit92].

In a general framework for abstracting transition systems that accommodates for the preservation of universal as well as existential temporal properties, not only the evaluation of atomic propositions in states, but also the treatment of transitions between states becomes non-standard. Notions of abstract transition systems that feature two different, dual transition relations are presented in [CIY94, DGG94, GHJ01, Kel95], and the approach in [LGS$^+$95] uses two separate transition systems — intuitively, one representing an over- and the other an under-approximation. *Modal transition systems* ([LT88]) also combine two transition relations ("may" and "must") but there they are not strictly dual.

An orthogonal duality is formed by the distinction between *invariance* and *progress* properties. Although both are preserved in most of the frameworks mentioned above, abstraction tends to introduce more false counterexamples to progress than to safety properties. In terms of Floyd-Hoare style correctness proofs, abstractions tend to be more like *invariants* than *ranking functions*. This problem is addressed in [BLS00, CS01, DGG00b].

The question whether a finite abstraction that is suitable for model checking any given temporal property always exists, is answered positively in [KPV99].

4 Techniques/Algorithms

Abstraction techniques are the methods or algorithms that can be employed to construct abstract systems from concrete ones. One approach consist in having the user choose *abstract interpretations*, given a concrete system and a property to be verified. These are replacements of data types with smaller-sized types that only reflect certain aspects of the original values; operations on these types will then have to be lifted correspondingly. Such abstracted data types may already exists, e.g. in the form of a library, or they may be newly constructed ([DHJ$^+$01, dMGM99]). In the latter case, *safety* of the abstractions may have to be proven ([SBLS99]).

More ambitious are the attempts to automatically derive suitable abstractions, e.g. [ASSSV94, BLO98, CU98, DGG93, GS97, NK00, RS99]. The technique proposed in [GS97] is now known as *predicate abstraction* and has inspired many case studies, tools, and approaches to abstraction refinement, see e.g. [AKN02, BHPV00, BMMR01, BPR, CGJ$^+$00, DDP99, GQ01].

On the other hand there are several techniques that are less general but fully automatic, like slicing ([HDZ00]), variable hiding ([BH99, DHH02]), and localization reduction ([Kur94]).

5 Tools

Some tools that combine model checking with abstraction and the URLs at which they can be found are:

α**Spin**: http://polaris.lcc.uma.es/˜gisum/fmse/tools/
Bandera: http://www.cis.ksu.edu/santos/bandera/
SLAM: http://www.research.microsoft.com/projects/slam/
FeaVer: http://cm.bell-labs.com/cm/cs/what/feaver/
InVeSt: http://www-verimag.imag.fr/˜async/INVEST/
JPF: http://ase.arc.nasa.gov/visser/jpf/
STeP: http://www-step.stanford.edu/

References

[AKN02] Nina Amla, Robert P. Kurshan, and Kedar S. Namjoshi. AutoAbs: Syntax-directed program abstraction, 2002. Submitted.

[ASB+94] Adnan Aziz, Vigyan Singhal, Felice Balarin, Robert K. Brayton, and Alberto L. Sangiovanni-Vincentelli. Equivalences for fair Kripke structures. In Serge Abiteboul and Eli Shamir, editors, *Automata, Languages and Programming*, number 820 in LNCS, pages 364–375, Springer-Verlag, Berlin, 1994.

[ASSSV94] Adnan Aziz, Thomas R. Shiple, Vigyan Singhal, and Alberto L. Sangiovanni-Vincentelli. Formula-dependent equivalence for compositional CTL model checking. In David L. Dill, editor, *Computer Aided Verification*, number 818 in LNCS, pages 324–337, Springer-Verlag, Berlin, 1994.

[BCG88] M.C. Browne, E.M. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Journal of Theoretical Computer Science*, 59:115–131, 1988.

[BFG+91] A. Bouajjani, J.C. Fernandez, S. Graf, C. Rodriguez, and J. Sifakis. Safety for branching time semantics. In J. Leach Albert, B. Monien, and M. Rodríguez Artalejo, editors, *Automata, Languages and Programming*, number 510 in LNCS, pages 76–92, Springer-Verlag, New York, 1991.

[BFH+92] A. Bouajjani, J.-C. Fernandez, N. Halbwachs, P. Raymond, and C. Ratel. Minimal state graph generation. *Science of Computer Programming*, 18:247–269, 1992.

[BG99] Glenn Bruns and Patrice Godefroid. Model checking partial state spaces with 3-valued temporal logics. In Halbwachs and Peled [HP99], pages 274–287.

[BH99] Ramesh Bharadwaj and Constance L. Heitmeyer. Model checking complete requirements specifications using abstraction. *Automated Software Engineering: An International Journal*, 6(1):37–68, January 1999.

[BHPV00] G. Brat, K. Havelund, S. Park, and W. Visser. Model checking programs. In *IEEE International Conference on Automated Software Engineering (ASE)*, 2000.

[BLO98] Saddek Bensalem, Yassine Lakhnech, and Sam Owre. Computing abstractions of infinite state systems compositionally and automatically. In Hu and Vardi [HV98], pages 319–331.

[BLS00] Kai Baukus, Yassine Lakhnech, and Karsten Stahl. Verifying universal properties of parameterized networks. In M. Joseph, editor, *Proceedings of the Sixth International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT2000*, number 1926 in LNCS, pages 291–303, Springer, Berlin, 2000.

[BMMR01] Thomas Ball, Rupak Majumdar, Todd Millstein, and Sriram K. Rajamani. Automatic predicate abstraction of C programs. *SIGPLAN Notices*, 36(5):203–213, 2001.

[BPR] Thomas Ball, Andreas Podelski, and Sriram K. Rajamani. Relative completeness of abstraction refinement for software model checking. To appear in TACAS 2002.

[BR83] Stephen D. Brookes and William C. Rounds. Behavioural equivalence relations induced by programming logics. In J. Diaz, editor, *Automata, Languages and Programming*, number 154 in LNCS, pages 97–108, Springer-Verlag, Berlin, 1983.

[BR01] Thomas Ball and Sriram K. Rajamani. Automatically validating temporal safety properties of interfaces. In Matthew Dwyer, editor, *Model Checking Software*, number 2057 in LNCS, pages 103–122, Springer, Berlin, 2001.

[Buc60] J. Buchi. Weak second-order arithmetic and finite automata. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, 1990.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. 4th ACM Symp. on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977.

[CDE+01] Marsha Chechik, Benet Devereux, Steve Easterbrook, Albert Y. C. Lai, and Victor Petrovykh. Efficient multiple-valued model-checking using lattice representations. In K. G. Larsen and M. Nielsen, editors, *International Conference on Concurrency Theory*, number 2154 in LNCS, pages 441–455, Springer, Berlin, 2001.

[CGJ+00] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In Emerson and Sistla [ES00], pages 154–169.

[CGL94] E.M. Clarke, O. Grumberg, and D.E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, September 1994.

[CGP99] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, Cambridge, Mass., 1999.

[Cho95] Ching-Tsun Chou. A simple treatment of property preservation via simulation. Technical Report 950014, Comp. Sc. Dept., University of California at Los Angeles, March 1995.

[CIY94] R. Cleaveland, S. P. Iyer, and D. Yankelevich. Abstractions for preserving all CTL^* formulae. Technical Report 94-03, Dept. of Comp. Sc., North Carolina State University, Raleigh, NC 27695, April 1994.

[CS01] Michael Colon and Henny Sipma. Synthesis of linear ranking functions. In Margaria and Yi [MY01], pages 67–81.

[CU98] Michael Colon and Tomas E. Uribe. Generating finite-state abstractions of reactive systems using decision procedures. In Hu and Vardi [HV98], pages 293–304.

[Dam96] Dennis René Dams. *Abstract Interpretation and Partition Refinement for Model Checking*. PhD thesis, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, July 1996.

[DDP99] Satyaki Das, David L. Dill, and Seungjoon Park. Experience with predicate abstraction. In Halbwachs and Peled [HP99], pages 160–171.

[DGG93] Dennis Dams, Rob Gerth, and Orna Grumberg. Generation of reduced models for checking fragments of CTL. In Costas Courcoubetis, editor, *Computer Aided Verification*, number 697 in LNCS, pages 479–490, Springer-Verlag, Berlin, 1993.

[DGG94] Dennis Dams, Orna Grumberg, and Rob Gerth. Abstract interpretation of reactive systems: Abstractions preserving $\forall$CTL*, $\exists$CTL* and CTL*. In E.-R. Olderog, editor, *Proceedings of the IFIP WG2.1/WG2.2/WG2.3 Working Conference on Programming Concepts, Methods and Calculi (PROCOMET)*, IFIP Transactions, Amsterdam, June 1994. North-Holland/Elsevier.

[DGG00a] Dennis Dams, Rob Gerth, and Orna Grumberg. Fair model checking of abstractions (extended abstract). In Michael Leuschel, Andreas Podelski, C.R. Ramakrishnan, and Ulrich Ultes-Nitsche, editors, *Proceedings of the Workshop on Verification and Computational Logic (VCL'2000)*, number DSSE-TR-2000-6, University of Southampton, July 2000.

[DGG00b] Dennis Dams, Rob Gerth, and Orna Grumberg. A heuristic for the automatic generation of ranking functions. In Ganesh Gopalakrishnan, editor, *Workshop on Advances in Verification (WAVe'00)*, pages 1–8, School of Computing, university of Utah, July 2000.

[DGLM99] Dennis Dams, Rob Gerth, Stefan Leue, and Mieke Massink, editors. *Theoretical and Practical Aspects of SPIN Model Checking*, number 1680 in LNCS, Springer, Berlin, 1999.

[DHH02] Dennis Dams, William Hesse, and Gerard Holzmann. Abstracting C with abC, 2002. Submitted.

[DHJ+01] Matthew Dwyer, John Hatcliff, Roby Joehanes, Shawn Laubach, Corina Pasareanu, Robby, Willem Visser, and Hongjun Zheng. Tool-supported program abstraction for finite-state verification. In *Proceedings of the 23rd International Conference on Software Engineering*, Toronto, Canada, May 12-19 2001. ICSE 2001, IEEE Computer Society.

[dMGM99] Maria del Mar Gallardo and Pedro Merino. A framework for automatic construction of abstract promela models. In Dams et al. [DGLM99], pages 184–199.

[DN87] Rocco De Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 24:211–237, 1987.

[DNV90] Rocco De Nicola and Frits Vaandrager. Three logics for branching bisimulation. In *1990 IEEE Fifth Annual Symposium on Logic in Computer Science*, pages 118–129, Los Alamitos, CA, 1990. IEEE Computer Society Press.

[ES00] E. Allen Emerson and A. Prasad Sistla, editors. *Computer Aided Verification*, number 1855 in LNCS, Springer, Berlin, 2000.

[Fit91] Melvin Fitting. Many-valued modal logics. *Fundamenta Informaticae*, 15(3–4):335–3, 1991.

[Fit92] Melvin C. Fitting. Many-valued modal logics II. In A. Nerode and M. Taitslin, editors, *Proc. LFCS'92*, number 620 in LNCS. Springer-Verlag, 1992.

[GHJ01] Patrice Godefroid, Michael Huth, and Radha Jagadeesan. Abstraction-based model checking using modal transition systems. In K. G. Larsen and M. Nielsen, editors, *International Conference on Concurrency Theory*, number 2154 in LNCS, pages 426–440, Springer, Berlin, 2001.

[Gin68] A. Ginzburg. *Algebraic Theory of Automata*. ACM Monograph Series. Academic Press, New York/London, 1968.

[GKP92] Ursula Goltz, Ruurd Kuiper, and Wojciech Penczek. Propositional temporal logics and equivalences. In W.R. Cleaveland, editor, *CONCUR '92*, number 630 in LNCS, pages 222–236, Springer-Verlag, Berlin, 1992.

[GQ01] R. Giacobazzi and E. Quintarelli. Incompleteness, counterexamples and refinements in abstract model-checking. In P. Cousot, editor, *Proc. of The 8th International Static Analysis Symposium, SAS'01*, volume 2126 of *Lecture Notes in Computer Science*, pages 356–373. Springer-Verlag, 2001.

[GS84] S. Graf and J. Sifakis. A modal characterization of observational congruence on finite terms of CCS. In Jan Paredaens, editor, *Proc. of the Eleventh International Colloquium on Automata Languages and Programming (ICALP)*, number 172 in LNCS, pages 222–234, Springer-Verlag, Berlin, 1984.

[GS97] S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In Orna Grumberg, editor, *Computer Aided Verification*, number 1254 in LNCS, pages 72–83, Springer, Berlin, 1997.

[GV90] Jan Friso Groote and Frits Vaandrager. An efficient algorithm for branching bisimulation and stuttering equivalence. In M. S. Paterson, editor, *Automata, Languages and Programming*, number 443 in LNCS, pages 626–638, Springer-Verlag, New York, 1990.

[GW89] R. J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G. X. Ritter, editor, *Information Processing 89*, pages 613–618, Amsterdam, 1989. North-Holland.

[HDZ00] John Hatcliff, Matthew B. Dwyer, and Hongjun Zheng. Slicing software for model construction. *Higher-Order and Symbolic Computation*, 13(4):315–353, 2000.

[Hei98] Constance L. Heitmeyer. On the need for practical formal methods. In A.P. Ravn and H. Rischel, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, number 1486 in LNCS, pages 18–26, Springer, Berlin, 1998.

[HJS01] Michael Huth, Radha Jagadeesan, and David A. Schmidt. Modal transition systems: A foundation for three-valued program analysis. In D. Sands, editor, *Programming Languages and Systems*, number 2028 in LNCS, pages 155–169, Springer, Berlin, 2001.

[HM80] Matthew Hennessy and Robin Milner. On observing nondeterminism and concurrency. In J.W. de Bakker and J. van Leeuwen, editors, *Proc. of the Seventh International Colloquium on Automata Languages and Programming (ICALP)*, number 85 in LNCS, pages 299–309, Springer-Verlag, Berlin, 1980.

[Hol01] G.J. Holzmann. From code to models. In *Proc. 2nd Int. Conf. on Applications of Concurrency to System Design*, pages 3–10, Newcastle upon Tyne, U.K., June 2001.

[Hop71] John Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Zvi Kohavi and Azaria Paz, editors, *Theory of Machines and Computations*, pages 189–196, Academic Press, New York, 1971.

[HP99] Nicolas Halbwachs and Doron Peled, editors. *Computer Aided Verification*, number 1633 in LNCS, Springer, Berlin, 1999.

[HS02] G.J. Holzmann and Margaret H. Smith. An automated verification method for distributed systems software based on model extraction. *IEEE Trans. on Software Engineering*, 28(4), April 2002.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts, 1979.

[HV98] Alan J. Hu and Moshe Y. Vardi, editors. *Computer Aided Verification*, number 1427 in LNCS, Springer, Berlin, 1998.

[Kel95] Peter Kelb. *Abstraktionstechniken für automatische Verifikationsmethoden*. PhD thesis, Carl von Ossietzky University of Oldenburg, Germany, December 1995.

[KPV99] Y. Kesten, A. Pnueli, and M. Vardi. Verification by augmented abstraction: The automata-theoretic view. In *Proceedings of the Annual Conference of the European Association for Computer Science Logic (CSL-99)*, LNCS, pages 307–321, Springer, Berlin, 1999.

[KS90] P.C. Kanellakis and S.A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86:43–68, 1990.

[Kur94] R. Kurshan. *Computer-aided Verification of Coordinating Processes: The Automata-Theoretic Approach*. Princeton University Press, 1994.

[LBBO01] Y. Lakhnech, S. Bensalem, S. Berezin, and S. Owre. Incremental verification by abstraction. In Margaria and Yi [MY01], pages 98–112.

[LGS+95] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6:11–44, January 1995.

[LT88] Kim G. Larsen and Bent Thomsen. A modal process logic. In *1988 IEEE Symposium on Logic in Computer Science*, pages 203–210, Computer Society Press, Washington, 1988.

[Mil71] R. Milner. An algebraic definition of simulation between programs. In *Second International Joint Conference on Artificial Intelligence*, pages 481–489, British Computer Society, London, 1971.

[Mil80] R. Milner. *A Calculus of Communicating Systems*. Number 92 in LNCS. Springer-Verlag, Berlin, 1980.

[MY01] Tiziana Margaria and Wang Yi, editors. *Tools and Algorithms for the Construction and Analysis of Systems*, number 2031 in LNCS, Springer, Berlin, 2001.

[NK00] Kedar S. Namjoshi and Robert P. Kurshan. Syntactic program transformations for automatic abstraction. In Emerson and Sistla [ES00], pages 435–449.

[NNH99] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer, Berlin, 1999.

[Par81] D. Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *Theoretical Computer Science*, number 104 in LNCS, pages 167–183, Springer-Verlag, Berlin, 1981.

[PT87] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computation*, 16(6):973–989, 1987.

[RS99] Vlad Rusu and Eli Singerman. On proving safety properties by integrating static analysis, theorem proving and abstraction. In W. Rance Cleaveland, editor, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS '99)*, number 1579 in LNCS, pages 178–192, Springer, Berlin, 1999.

[SBLS99] K. Stahl, K. Baukus, Y. Lakhnech, and M. Steffen. Divide, abstract, and model-check. In Dams et al. [DGLM99].

[Sti89] Colin Stirling. Comparing linear and branching time temporal logics. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification*, number 398 in LNCS, pages 1–20, Springer-Verlag, Berlin, 1989.

[vBvES94] Johan van Benthem, Jan van Eijck, and Vera Stebletsova. Modal logic, transition systems and processes. *Journal of Logic and Computation*, 4(5):811–855, 1994.

[vG90] R.J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions*. PhD thesis, Free University of Amsterdam/Center for Math. and Comp. Sc., Amsterdam, 1990.

[WC99] Andre Wong and Marsha Chechik. Formal modeling in a commercial setting: A case study. In *FM'99 - Formal Methods*, number 1708 in LNCS, pages 590–607, Springer, Berlin, 1999.