Chapter 9

Conclusion and future work

We have studied the specification and verification of some classes of reactive systems, namely discrete systems, real-time systems and parameterized systems. We use TLA* from MERZ to formalize our approach.

The general formula for representing reactive systems is as follows:

$$\exists\, x : \mathit{Init} \wedge \Box[\mathit{Next}]_v \wedge L$$

where

- x is a list of internal variable,
- *Init* is a state predicate that describes the initial states,
- *Next* is an action characterizing the system's next-state relation,
- v is a state function, and
- L is a formula stating the liveness conditions expected from the system.

This formula essentially describes a state machine, augmented by liveness condition, that generates the allowed behaviors of the system under specification. For the more specific classes of reactive systems, in particular the classes of reactive systems we have considered in this thesis:

- For discrete systems, a specification is a formula of the form $\mathit{Spec} \equiv \mathit{Init} \wedge \Box[\mathit{Next}]_v \wedge L_f$, where L_f is a conjunction of formula $\mathrm{WF}_v(A)$ and $\mathrm{SF}_v(A)$ where A is an action which appears as disjunct of *Next*.
- For real-time systems, a specification is a formula of the form $\mathit{RTSpec} \equiv \mathit{Init} \wedge \Box[\mathit{Next}]_v \wedge \mathit{RTNow}(v) \wedge \mathit{RT}$ where
  - $\mathit{RTNow}(v)$ is the formula that asserts that *now* (the variable used to model real-time) is initially equal to 0 and it increases monotonically and without bound and

  - RT is a conjunction of real-time bound formulas $RTBound(A_i, v, t_i, d_i, e_i)$ where A_i is a sub-action or disjunct of $Next$, t_i, d_i and e_i is the timer, lower bound and upper bound of A_i, respectively.

- For parameterized systems, a specification is a formula of the form $parSpec \equiv Init \wedge \Box[\exists\, k \in M : Next(k)]_v \wedge \forall\, k \in M : L_f(k)$.

In our methodology, we use a class of diagrams called predicate diagrams as abstract representation of the discrete systems being considered. Assume given a specification of discrete system *Spec* and a temporal formula F, the verification of discrete systems using our diagrams can be done in two steps:

- The first step is to find a diagram that conforms *Spec*. To prove that a diagram conforms to a specification, we equip the diagram with a corresponding conformance theorem in order to produce some proof obligations. The proof is done deductively either manually by hand or by using an automatic theorem prover.

- The second step is to prove that all traces through the diagram satisfy F. In this step, we view the diagram as a finite transition system that is amenable to model checking. All predicates and actions that appear as labels of nodes or edges are then viewed as atomic propositions. Regarding predicate diagrams as finite labeled transition systems, their runs can be encoded in the input language of standard model checkers such as SPIN.

Thus, our methodology can be viewed as an integration between deductive and algorithmic verification techniques.

In Section 5.6, we have successfully proven the completeness of predicate diagram. The proof is done in four steps:

- The construction of formula automaton $\mathcal{M}^f$ which is a Muller automaton accepting exactly the behaviors satisfying F.

- The construction of specification automaton $\mathcal{M}^s$, which is a Muller automaton such that the accepting condition is defined in a way such that it exactly characterizes the fairness of *Spec*.

- The construction of product automaton $\mathcal{M}^p$, which is the product automaton of $\mathcal{M}^f$ and $\mathcal{M}^s$. Thus, the properties of $\mathcal{M}^p$ are inherited from $\mathcal{M}^f$ and $\mathcal{M}^s$.

- The last step is the translation of the product automaton into predicate diagram.

We have also shown that the concept of predicate diagrams is capable enough to handle some other classes of reactive systems such as real-time systems and parameterized systems. To verify real-time systems, we define a variant of predicate diagrams called *timed predicate diagrams* or TPDs. The idea of these diagrams is to use the components of predicate diagrams related to discrete properties and to replace the components related to the fairness conditions with some components related to real time conditions. For the components related to real-time property, we adopt the structure of timed-automata. Thus, in one direction, TPDs can be viewed as an extension of predicate diagrams. In the other direction, we may say that predicate diagrams are restricted TPDs. Particularly, when we eliminate all the components of timed predicate diagrams that are related to real-time property, then we have predicate diagrams without fairness conditions. We call such a predicate diagram the *untimed version* of a TPD. In the context of parameterized systems, we have shown that the (ordinary) predicate diagrams can still be used for proving the properties that are related to the whole processes. Whereas to prove the universal properties, i.e. the properties that are related to one single process, we define a class of diagrams called *parameterized predicate diagrams* or PPDs.

The verification of real-time systems and parameterized systems using TPDs and PPDs are similar to the verification of discrete systems using predicate diagrams.

Using the concept of abstract interpretation we have shown that our diagrams can be generated semi-automatically. We use the term "semi-automatically", since the user's intervention is still needed, in particular for defining the abstraction functions and rewriting rules. We have developed two prototype tools: PreDiaG, for the generation of predicate diagrams, and parPreDiaG, for the generation of PPDs.

Some possible tracks for future work that come to mind are listed below.

- **Hybrid systems.** Basically, hybrid systems can be viewed as the union of discrete and real-time systems. However, it is still needed to study the special characteristic of this class of systems and to investigate the extension or modification that should be done over predicate diagrams.

- **Completeness of TPDs and PPDs.** In this work, we only consider the completeness of predicate diagrams. The proof of the completeness

of TPDs and PPDs should be an interesting topic for a research. For proving the completeness of TPDs, it is indicated, that we can use the concept of timed automata and do the similar proof as we did in proving the completeness of predicate diagrams. However, this indication is still needed to be explored. Unfortunately, the proof of PPDs is still an open question.

- **Tool support.** For the practical application of our method tool support is essential. The tools we have implemented are prototypes that are still needed to be improved, in particular in the aspect of graphical user interface. We have shown that the generation of diagrams can be done incrementally. We should or may refine the diagrams resulted by our tools, until we get the desired diagrams. Thus, there is also a need to have a good graphical editor that can support the refinement of the diagrams. It is also desirable to have a translator from TLA$^+$ to MONA syntax and to integrate these tools with an existing automatic theorem prover in order to prove the proof obligations whenever needed.

Bibliography

[1] Martin Abadi and Leslie Lamport. An old-fashioned recipe for real time. *ACM Transactions on Programming Languages and Systems*, 16(5):1543-1571, September 1994.

[2] Martin Abadi and Stephan Merz. On TLA as a logic. In Manfred Broy, editor, Deductive Program Design, NATO ASI series F, pages 235-272, Springer-Verlag, Berlin, 1996.

[3] M.W. Alford, J.P. Ansart, G. Hommel, L. Lamport, B. Liskov, G.P. Mullery and F.B. Schneider. Distributed Systems: Methods and tools for specification. Volume 190 of *Lecture Notes in Computer Science*. Springer-Verlag, 1985.

[4] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21:181-185, October 1985.

[5] Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. Technical Report 86-727, Cornell University, Ithaca, New York, January 1986.

[6] Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. *Distributed Computing 2*, pp. 117-125, 1987.

[7] Rajeev Alur. Timed automata. NATO ASI Summer School on Verification of Digital and Hybrid Systems, 1998.

[8] Rajeev Alur, C. Courcoubetis and David L. Dill. Model-checking for real-time systems. In *Proceeding of the 5th Annual Symposium on Logic in Computer Science*, pp 414-425. *IEEE Computer Society Press*, 1990.

[9] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science* 126:183-235, 1994.

[10] R. Alur and T.A. Henzinger. A really temporal logic. In *Proc. 30th IEEE Symp. Found. of Comp. Sci.*, pages 164-169, 1989.

[11] R. Alur and T.A. Henzinger. Logics and models of real time: A survey. In J.W. de Bakker, C. Huizing, W.P. de Roever and G. Rozember, editors, *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, volume 600 of *Lecture Notes in Computer Science*, pages 74-106. Springer-Verlag, 1992.

[12] K. Apt and D. Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, Volume 15, pp. 307-309. 1986.

[13] Kai Baukus, Yassine Lakhnech and Karsten Stahl. Verifying Universal Properties of Parameterized Networks. Technical Report TR-sT-00-4, CAU Kiel, July, 2000.

[14] Kai Baukus, Saddek Bensalem, Yassine Lakhnech and Karsten Stahl. Abstracting WS1S Systems to Verify Parameterized Networks. In *Proceeding of the 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000)*, Volume 1785 of *Lecture Notes in Computer Science*, pages 188-203. Springer, 2000.

[15] J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson, Y. Wang and Carsten Weise. New Generation of UPPAAL. *Int. Workshop on Software Tools for Technology Transfer.* June 1998.

[16] S. Bensalem, Y. Lakhnech and S. Owre. Computing abstractions of infinite state systems automatically and compositionally. In *Conference on Computer Aided Verification (CAV-98)*, volume 1427 of *Lecture Notes in Computer Science*, pages 319-331. Springer-Verlag, 1998.

[17] S. Bensalem, et.al. An overview of SAL. In C M. Holloway, editor, *LFM 2000: 5th NASA Langley Formal Methods Workshop*, pages 187-196, 2000.

[18] M. Bozzano and G. Delzanno. Beyond Parameterized Verification. In *Proceedings of International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2002.* Volume 2280 of *Lecture Notes in Computer Science*, pages 221-235. Springer, 2002.

[19] Anca Browne, Luca de Alfaro, Zohar Manna, Henny B. Sipma and Tomás Uribe. *Diagram-based Formalisms for the Verification of Reactive Systems.* In *CADE-13 Workshop on Visual Reasoning*, New Brunswick, NJ, July 1996.

[20] Anca Browne, Zohar Manna and Henny B. Sipma. Generalized verification diagrams. In *15th Conference in the Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *Lecture Notes in Computer Science*, pages 484-498, December, 1995.

[21] Anca Browne, Zohar Manna and Henny B. Sipma. Modular verification diagrams. Technical report Computer Science Departement, Stanford University, 1996.

[22] R.E. Bryant. Graph-based algorithmics for boolean function manipulation. *IEEE Transactions on Computers* C-35(8):677-691.

[23] J.R. Büchi. Weak second-order arithmatic and finite automata. Z. Math. Logik Grundl. Math., 6:66-92, 1960.

[24] J.R. Büchi. On a decision method in restricted second order arithmatic. *Proceedings of the International Congress on Logic, Method and Philysophy in Science* 1960, Stanford, CA, 1962. Stanford University Press, 1-12.

[25] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill and L.J. Hwang. Symbolic model checking:10^20 states and beyond. *Information and Computation* 98(2):142-170.

[26] Dominique Cansell, Dominique Méry and Stephan Merz. Predicate diagrams for the verification of reactive systems. In 2^{nd} *Intl. Conf. on Integrated Formal Methods (IFM 2000)*, vol. 1945 of *Lectures Notes in Computer Science*, Dagstuhl, Germany, November 2000. Springer-Verlag.

[27] E.M. Clarke and E.A. Emerson. Characterizing correctness properties of parallel programs using fixpoints. *International Colloquim on Automata, Languages and Programming.* Vol. 85 of *Lecture Nodes in Computer Science*, pp. 169-181, Springer-Verlag, July, 1980.

[28] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. *Workshop on Logic*

of Programs, Yorktown Heights, NY. Vol. 131 of *Lecture Nodes in Computer Science*, pp. 52-71, Springer-Verlag, 1981.

[29] E.M. Clarke, T. Filkorn and S. Jha. Exploiting symmetry in temporal logic model checking. In Courcoubetis, editor. *Proceedings of the 5th Workshop on Computer-Aided Verification.* Volume 693 of *Lecture Nodes in Computer Science*, pp. 450-462. Springer, 1993.

[30] E.M. Clarke and O. Grumberg. Avoiding the state explosion problem in temporal logic model checking. Proceedings of the 6th annual ACM Symposium on Principles of Distributed Computing, pp. 294 - 303. Columbia, Canada, August 1987.

[31] E.M. Clarke, O. Grumberg and D.E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5), 1994.

[32] Edmund M. Clarke, Orna Grumberg and Doron A. Peled. Model Checking. The MIT Press, 1999.

[33] M.A. Colón and T.E. Uribe. Generating finite-state abstraction of reactive systems using decision procedures. In *Conference on Computer-Aided Verification*, volume 1427 of *Lecture Notes in Computer-Science*, pages 293-304. Springer-Verlag, 1998.

[34] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th ACM Symp. Princ. of Prog. Lang.*, pp. 238-252. ACM Press, 1977.

[35] Radhia Cousot. Fondements de méthodes de preuve d'invarince et de fatalité de programmes paralléles. PhD thesis. INPL, 1985.

[36] D. Dams. Abstract interpretation and partition refinement for model checking. PhD thesis, Technical University of Eindhoven, 1996.

[37] D. Dams, R. Gerth and O. Grumberg. Abstract interpretation of reactive systems: Abstractions preserving ACTL*, ECTL* and CTL*. In *Proceedings of the IFIP WG2.1/WG2.2/WG2.3 (PROCOMET).* IFIP Transactions, North-Holland/Elsevier, 1994.

[38] D. Dams, R. Gerth and O. Grumberg. A heuristic for the automatic generation of ranking functions. In *Proceedings of Workshop on Advances in Verification*, pages 1-8. 2000.

[39] Marco Daniele, Fausto Giunchiglia and Moshe Y. Vardi. Improved Automata Generation for Linear Temporal Logic. In *Proc. 11th Intl. Conference on Computer Aided Verification.* Volume 1633 of *Lecture Notes in Computer Science*, pages 249-260. Springer, 1999.

[40] S. Das, D.L. Dill and S. Park. Experience with predicate abstractions. In *Proc. 11th Intl. Conference on Computer Aided Verification.* Volume 1633 of *Lecture Notes in Computer Science.* pages 160-171, Springer, 1999.

[41] L. de Alfaro and Zohar Manna. Temporal verification by diagram transformations. In *Proc. 8th International Conference on Computer Aided Verification.* Volume 1102 of *Lecture Notes in Computer Science*, pages 288-299. Springer, July, 1996.

[42] D. Detlefts, G. Nelson, and J. Saxe. Simplify: the ECS theorem prover. Technical report, Systems Research Center, Digital Equipment Corporation, Palo Alto, CA, November 1996.

[43] E.A. Emerson and A.P. Sistla. Symmetry and model checking. In Courcoubetis, editor. *Proceedings of 5th Workshop on Computer-Aided Verification*, pp. 463-478. June/July 1993.

[44] E.A. Emerson and K.S. Namjoshi. Automatic verification of parameterized synchronous systems. In *Proceeding of 8th Conference on Computer-Aided Verification.* Volume 1102 of Lecture Notes in Computer Science, pp. 87-98. Springer, 1996.

[45] E.A. Emerson and K.S. Namjoshi. Verification of a parameterized bus arbitration protocol. Volume 1427 of *Lecture Notes in Computer Science*, pp. 452–463. Springer,1998.

[46] Melvin Fitting. First-order logic and automated theorem proving. *Graduate Texts in Computer Science.* Springer-Verlag. 1996.

[47] Rober W. Floyd. Assigning meanings to programs. *Proc. Symposia in Applied Mathematics*, 19:19-32, 1967.

[48] Jean H. Gallier. Logic for Computer Science: Foundation of automatic theorem proving. Harper & Row, Publisher, Inc. New York. 1986.

[49] Paul Gastin and Denis Oddoux. Fast LTL to Buchi Automata Translation. Proceedings of *13th Conference on Computer-Aided Verification.* Volume 2102 of *Lecture Notes in Computer Science*, pages 53-65. Springer, 2001.

[50] S. German and A.P. Sistla. Reasoning about systems with many processes. Journal of the ACM, Vol. 39, Number 3, July 1992.

[51] Rob Gerth, Doron Peled, Moshe Y. Vardi and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. PSTV 1995: 3-18.

[52] P. Godefroid and D. Pirottin. Refining dependencies improves partial-order verification methods. In *Proceedings of the 5th Conference on Computer-Aided Verification.* Volume 697 of *Lecture Notes in Computer Science*, pp. 438-449. Springer, 1993.

[53] S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In O. Grumberg, editor, *Conference on Computer Aided Verifications.* Volume 1254 of *Lecture Notes in Computer-Science*, pp. 72-83. Springer-Verlag, 1997. June 1997, Haifa, Israel.

[54] K. Havelund and N. Shankar. Experiments in theorem proving and model checking for protocol verification. FME. Volume 1051 of *Lecture Notes in Computer Science*, pages 662-681. Springer, 1996.

[55] T. Henzinger, Z. Manna, and A. Pnueli. Temporal Proof Methodologies for Timed Transition Systems. *Information and Computation*, 112(2):273-337, 1994.

[56] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576-580, 1969.

[57] G. Holzmann. The SPIN model checker. *IEEE Trans. on software engineering*, 16(5):1512-1542. May 1997.

[58] Y. Kesten and A. Pnueli. Taming the Infinite: Verification of Infinite-State Reactive Systems by Finitary Means. In *Engineering Theories of Software Construction, (NATO) Science Series*, Series III: Computer and Systems Sciences, Vol. 180, pages 261-299, IOS Press 2001.

[59] Y. Kesten, Z. Manna and A. Pnueli. Verification of Clocked and Hybrid Systems. In G. Rozenberg and F.W. Vaandrager, editors, Lectures on

Embedded Systems, volume 1494 of *Lecture Notes in Computer Science*, pages 4-73. Springer-Verlag, 1998.

[60] E. Kindler. Safety and Liveness Properties: A survey. *Bulletin of the European Association for Theoretical Computer Science*, Vol. 53, pp. 268-272, 1994.

[61] N. Klarlund and A. Møller. MONA Version 1.3 UserManual. BRICS, 1998.

[62] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems*, 2(4):255-299, 1990.

[63] R. Koymans and W.-P. Roever. Examples of a real-time temporal logic specifications. In B.D. Denvir, W.T. Harwood, M.I. Jackson and M.J. Wray, editors, *The analysis of concurrent systems.* Volume 207 of *Lecture Notes in Computer Science*, pages 231-252. Springer-Verlag, 1985.

[64] R. Koymans, J. Vytopyl and W.-P. de Roever. Real-time programming and asynchronous message passing. In *Proc. 2nd ACM Symp. Princ. of Dist. Comp.*, pages 187-197,1983.

[65] Fred Kröger. Temporal logic of programs. EATCS Monographs on Theoretical Computer Science, Vol. 8. Springer-Verlag. 1986.

[66] Robert P. Kurshan. Computer Aided Verification of Coordinating Processes:The automata-theoretic approach. Princeton University Presss. 1994.

[67] Leslie Lamport. A new solution of Dijkstra's concurrent programming problem. *Communications of the ACM*, 17(8):435-455, 1974.

[68] Leslie Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, SE-3(2):125-143, March, 1977.

[69] Leslie Lamport. The Temporal Logic of Actions. *ACM Transactions on Programming Languages and Systems*, 16(3) : 872-923, May 1994.

[70] Leslie Lamport. TLA in Pictures. *SRC Research Report 127*, Digital System Researh, California, 1994.

[71] Leslie Lamport. Introduction to TLA. *SRC Technical Node 1994-001*, Digital System Research, California. December, 1994.

[72] Leslie Lamport. Specifying concurrent systems with TLA$^+$. In *Calculation System Design.* M. Broy and R. Steinbrüggen, editors. IOS Press, Amsterdam, 1999.

[73] Leslie Lamport. Specifying Systems: The TLA+ Language and Tools or Hardware and Software Engineers. Addison-Wesley, 2002.

[74] C. Loiseaux, S. Graf, J. Sifakis, A. Boujjani and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6(1), 1995.

[75] Zohar Manna, Michael Colon, Bernd Finkbeiner, Henny Sipma and Tomás Uribe. Abstraction and Modular Verification of Infinite-State Reactive Systems. In *Requirements Targeting Software and Systems Engineering (RTSE).* Volume 1526 of *Lecture Notes in Computer-Science*, pp 273-292. Springer, 1998.

[76] Zohar Manna and Amir Pnueli. A Hierarchy of Temporal Properties. In *Proc. ACM Symposium on Principles of Distributed Computing*, 1990.

[77] Zohar Manna and Amir Pnueli. Models for reactivity. *Acta Informatica*, 30:609-678, 1993.

[78] Zohar Manna and Amir Pnueli. Clocked Transition Systems. Technical Report STAN-CS-TR-96-1566, Dept. of Computer Science, Stanford University. April, 1996.

[79] Zohar Manna and Amir Pnueli. Temporal verification diagrams. In *Proc. Intl. Symposium on Theoretical Aspects of Computer Software.* Volume 697 of *Lecture Notes in Computer Science*, pages 726-765. Springer-Verlag, 1994.

[80] Zohar Manna and Amir Pnueli. Verification of parameterized programs. In *Specification and Validation Methods (E. Borger, ed.)*, Oxford University Press, pp. 167-230, 1994.

[81] Zohar Manna and Amir Pnueli. Temporal verification of reactive systems: safety, Springer-Verlag New York, Inc., New York, NY, 1995.

[82] K.L. McMillan. Symbolic model checking:an approach to the state explosion problem. Kluwer Academic, 1993.

[83] Stephan Merz. Logic-based analysis of reactive systems: hiding, composition and abstraction. Habilitationsschrift. Institut für Informatik. Ludwig-Maximillians-Universitt, Munich Germany. December 2001.

[84] R. McNaughton. Testing and generating infinite sequence by a finite automaton. Inform. Contr. 9, pages 521-530, 1966.

[85] J. Misra and K.M. Chandy. Parallel program design: a foundation. Addison-Wesley Publishers, 1988.

[86] D.E. Muller. Infinite sequences and finite machines. In *Proc. 4th IEEE Symp. on Switching Circuit Theory and Logical Design*, 99:3-16, 1963.

[87] C.E. Nugraheni. Prediag: A tool for the generation of predicate diagrams. In *Proceeding of Student Research Forum, SOFSEM 2002*, pp. 35–40, November 2002.

[88] J.S. Ostroff. Formal methods for the specification and design of real-time safety critical systems. In *Journal of Systems and Software*, Vol. 18, Number 1, April 1992.

[89] J.S. Ostroff. Temporal logic of real-time systems. Advanced Software Development Series. Research Studies Press (John Wiley & Sons), Taunton, England, 1990.

[90] Susan Owicki and Leslie Lamport. Proving liveness properties of concurrent programs. *ACM Transactions on Programming Languages and Systems*, 4(3):455-495, July 1982.

[91] PAX Tool:Parameterized systems Abstracted and eXplored. Avaiable at `http://www.informatik.uni-kiel.de/~kba/pax/`.

[92] Doron Peled. Combining partial order reductions with on-the-fly model-checking. In Dill, editor. *Proceedings of the 1994 Workshop on Computer-Aided Verification.* Volume 818 of *Lecture Notes in Computer Science*, pages 377-390. Springer-Verlag, 1994.

[93] Doron Peled. Software reliability methods. Texts in Computer Science. Springer, 2001.

[94] Amir Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symposium Foundation of Computer Science*, pages 46-57, IEEE Computer Society Press, 1977.

[95] J.P. Quielle and J. Sifakis. Specification and verification of concurrent systems in CESAR. In M. Dezani-Cianzaglini and Ugo Montanari, editors, *International Symposium on Programming.* Volume 137 of *Lecture Notes in Computer Science*, pp. 337-350. Springer-Verlag, 1981.

[96] M.O. Rabin. Decidability of second-order theoried and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1-35, 1969.

[97] Fred B. Schneider. Decomposing properties into safety and liveness using predicate logic. Technical Report 87-874, Departement of Computer Science, Cornell University, Ithaca, New York, October 1987.

[98] H. Saidi and N. Shankar. Abstract and model check while you prove. In N. Halbwachs and D. Peled, editors, *Conference on Computer-Aided Verification (CAV'99).* Volume 1633 of *Lecture Notes in Computer-Science*, pages 443-454, Trento, Italy, 1999, Springer-Verlag.

[99] Henny B. Sipma. Diagram-based verification of discrete, reactive and hybrid systems. PhD Thesis, Dept. of Computer Science, Standford University, 1999.

[100] A.P. Sistla. On the characterization of safety and liveness properties in temporal logic. In *Proceeding of the 4th annual ACM Symposium on Principles of Distributed Computing*, pages 39-48, Minaki, Ontario, Canada, August, 1985. ACM.

[101] Fabio Somenzi and Roderick Bloem. Efficient Büchi Automata from LTL Formulae. In *the 12th Conference on Computer Aided Verification (CAV'00).* Volume 1633 of *Lecture Notes in Computer Science*, pages 247-263. Springer Verlag, 2000.

[102] STERIA - Technolgies de l'Information, Aix-en-Provence (F). *Atelier B, Manual Utilisateur*, 1998. Version 3.5.

[103] W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume II:Formal Methods and Semarntics*, pages 134-191. Elsevier Sciences Publishers B.V., 1990.

[104] TReX Examples: Fischer protocol in `http://www_verimag.imag.fr/~annichin/trex/demos/fischer.html`.

[105] A. Valmari. A stubborn attack on state explosion. In *Proceedings of the 2nd Workshop on Computer-Aided Verification.* Volume 663 of *Lecture Notes in Computer Science*, pages 260-175. Springer-Verlag, 1992.

[106] Pierre Wolper. Constructing automata from temporal logic formulas: A tutorial. In Lectures on Formal Methods in Performance Analysis (First EEF/Euro Summer School on Trends in Computer Science). Volume 2090 of *Lecture Notes in Computer Science*, pages 261-277. Springer-Verlag, July 2001.

[107] P. Wolper and V. Lovinfose. Verifying properties of large sets of processes with network invariants. In J. Sifakis (ed), *Automatic Verification Methods for Finite State Systems.* Volume 407 of *Lecture Notes in Computer Science*, pages 68-80. Springer-Verlag, 1990.

[108] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proceeding of the First Symposium on Logic in Computer Science*, pages 322-331. Cambridge, June 1986.

[109] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1-37, 1994.

[110] Sergio Yovine. Kronos: A verification tool for real-time. In *International Journal of Software Tools for Technology Transfer*, Vol. 1, Nber. 1+2, p.123-133, December 1997. Springer-Verlag Berlin Heidelberg 1997.