Course Outline, Readings, and Tools

This course will explore the available state-machine-based methods for modeling and analyzing complex systems and will attempt to determine what is needed to make them more useful in practice. Students will study and learn to use a broad set of modeling techniques and analysis tools, and will apply them to case studies in areas such as data management in distributed networks, reliable network communication, mobile computing, automated transportation, and object-oriented programming.

The course will be organized according to increasing complexity of system features to be modeled. We will start with basic, unstructured, untimed state machines, considering first safety properties and then liveness properties. We will proceed to consider composition, then add complications of timing, hybrid continuous/discrete behavior, and probability. At each step, we will study the basic mathematical foundations and study and use some of the available tools.

1 Simple Asynchronous Systems, Safety Properties

1.1 Mathematical Foundations

[3 classes] Leader: Nancy Lynch

The main ideas here are state machines, invariants, labeled state machines, I/O automata, abstraction functions, and abstraction relations. These are introduced in:

Nancy Lynch. 6.042 lecture notes, Fall 2000. `safety/6042-L08..9..10.ps`.

Nancy A. Lynch and Mark R. Tuttle. An introduction to input/output automata. *CWI-Quarterly*, 2(3):219–246, September 1989. Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands. Technical Memo MIT/LCS/TM-373, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, November 1988. `safety/CWI89.ps`.

Nancy Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, March 1996. Chapter 8.

Lamport describes a mathematical state-machine model with an associated logic (TLA). For now, we will consider the safety aspects of this work, then return to the liveness aspects later.

Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994. `safety/Lamport-tla-toplas.ps`.

Manna and Pnueli describe "reactive systems". Again, we will begin with safety aspects, returning to liveness later.

Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57, Providence, Rhode Island, October-November 1977. IEEE.

Amir Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

Z. Manna and A. Pnueli. Temporal verification diagrams. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Software*, volume 789 of *Lecture Notes in Computer Science*, pages 726–765. Springer-Verlag, April 1994. `safety/MannaP-diagrams.ps`.

Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, New York, 1992. The safety parts.

Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.

Various kinds of simulation relations and abstraction mappings are described in:

Nancy Lynch and Frits Vaandrager. Forward and backward simulations — Part I: Untimed systems. *Information and Computation*, 121(2):214–233, September 1995. `safety/IC95.ps`.

Willem-Paul de Roever and Kai Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison.* Cambridge University Press, 1998.

Henrik Ejersbo Jensen. *Abstraction-Based Verification of Distributed Systems.* PhD thesis, Department of Computer Science, Institute for Electronic Systems, Aalborg University, Aalborg, Denmark, June 1999. R-99-5005.

A hierarchical model for describing state machines and a corresponding visual formalism, appear in:

David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

D. Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, 1988. (Reprinted in *Diagrammatic Reasoning* (Chandrasekaran et al., eds), AAAI Press and MIT Press, 1995, pp. 235-271, and in *High Integrity System Specification and Design* (Bowen and Hinchey, eds.), Springer-Verlag, London, 1999, pp. 623-657.)

1.2 Tools

1.2.1 Languages and parsers

The IOA formal language for specifying input/output automata is described in:

Stephen J. Garland, Nancy A. Lynch, and Mandana Vaziri. IOA: A language for specifying, programming and validating distributed systems, December 2000. User and Reference Manual. `safety/ioaManual.ps`.

Stephen J. Garland and Nancy A. Lynch. *Using I/O Automata for Developing Distributed Systems*, chapter 13, pages 285–312. Cambridge University Press, USA, 2000. `safety/ioa-leavens.ps`.

The IOA language has a front end, which parses and checks static semantics. It also produces internal Java data structures representing the various parts of automata, but those aren't yet hooked up to usable validation tools. This represents work in progress.

[TOOL: IOA front end: parser and checker]
Students can find links to the manual plus other information about IOA and the IOA Toolkit in
`http://nms.lcs.mit.edu/~garland/IOA/`.
A direct link to the front end checker is `http://nms.lcs.mit.edu/~garland/IOA/ioaCheck.tar.gz.`

The TLA+ language is a formal specification language based on Lamport's TLA logic. This language is described in Lamport's new book, of which we have a draft version:

Leslie Lamport. Specifying concurrent systems with TLA+. Preliminary draft, December 2000. Chapters 1-7.

Descriptions written in TLA+ can be checked using the TLA+ parser:
[TOOL: TLA+ parser]

1.2.2 Model checkers

[3 classes] Leaders: Nancy Lynch, Daniel Jackson, Paul Attie

We will first focus on model-checking for safety properties, and later on liveness. An overview of the ideas in various model checkers appears in:

E. M. Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press, December 1999.

E. M. Clarke, O. Grumberg, and D. E. Long. Model checking. In *A survey on model checking, abstraction and composition*, volume 152 of *Nato ASI Series F*. Springer-Verlag, 1996. `safety/CGL-markt.ps`.

Symbolic model checking is discussed in:

K.L. McMillan. *Symbolic Model Checking: an Approach to the State Explosion Problem*. PhD thesis, Carnegie Mellon University, 1992. CMU-CS-92-131.

[TOOL: SMV] `http://sra.itc.it/tools/nusmv/`

Some selected basic model-checking papers are:

E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal-logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986. `http://theory.lcs.mit.edu/classes/6.897/papers/safety/p244-Clarke.pdf`.

E. Allen Emerson and A. Prasad Sistla. Symmetry and model checking. *Formal Methods in System Design*, 9(1/2):105-130, 1996. `safety/EmersonS-symmetry.ps`.

E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16(5):1512–, September 1994. `safety/CGL-abstraction.ps`.

The SPIN model checker is described in:

Gerard J. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall Software Series, New Jersey, 1991.

[TOOL: SPIN]

The TLC model-checker is described in:

Leslie Lamport. Specifying concurrent systems with TLA+ preliminary draft, December 2000. Chapters 12-13.

[TOOL: TLC]

1.2.3 Invariant discovery tools

[1 class] Leader: Michael Ernst

The notion of dynamic detection of likely invariants, and the Daikon tool for doing so, is described in:

Michael D. Ernst, Jake Cockrell, William G. Griswold, and David Notkin. Dynamically discovering likely program invariants to support program evolution. *IEEE Transactions on Software Engineering*, 27(2):1–25, February 2001. `safety/Ernst-invariants-tse.pdf`.

[TOOL: Daikon]

1.2.4 Theorem provers

[3 classes] Leader: Steve Garland

An overview of interactive theorem-proving ideas and tools will be given, with some discussion of the particular theorem prover LP and high-level pointers to other theorem provers like PVS and Isabelle. Some sources for overview material:

> Handbook of theoretical computer science: Volume b, formal models and semantics, 1990. Chapter 6, on term rewriting systems; Chapter 13, on algebraic specification; Chapters 14-16, on logics for proving things about programs.
>
> Stephen J. Garland and John V. Guttag. LP, the Larch Prover. `www.sds.lcs.mit.edu/~garland/LP/overview.html`..
>
> S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M. Srivas. PVS: Combining specification, proof checking, and model checking. In *CAV '96*, volume 1102 of *Lecture Notes in Computer Science*, pages 411–414. Springer Verlag, 1996. `safety/pvs-cav96.ps.gz`.
>
> N. Shankar, Sam Owre, and John Rushby. The PVS proof checker: A reference manual. Technical report, Computer Science Lab, SRI Intl., Menlo Park, CA, 1993. Also see the Language Reference, System Guide, and Prover Guide all available from `pvs.csl.sri.com/manuals.html`.
>
> L. C. Paulson. *Isabelle: A Generic Theorem-Prover.* LNCS 828. Springer-Verlag, 1994. `www.cl.cam.ac.uk/Research/HVG/Isabelle`.

We will learn one theorem prover: MIT's own Larch Prover:

[TOOL: Larch Prover]

`http://www.sds.lcs.mit.edu/~garland/LP/overview.html`

2 Simple Asynchronous Systems, Liveness Properties

[2 classes] Leaders: Nancy Lynch, Paul Attie

2.1 Foundations

Simple termination properties are described in:

> Nancy Lynch. 6.042 lecture notes, Fall 2000. Lectures 8, 9, 10. `safety/6042-L08..9..10.ps`.

Basic definitions of safety vs. liveness appear in:

> Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, October 1985.
>
> Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.

A treatment of fairness for I/O automata appears in:

> Nancy Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, March 1996. Chapter 8.

The following gives an overview of temporal logics, which are used to prove liveness properties:

E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Volume B, Formal Models and Semantics*, pages 995–1072. North Holland Pub. Co./MIT Press, 1990. Chapter 16.

The work by Manna and Pnueli, listed in Section 1.1, describes liveness features in temporal logics. We will revisit this here, and will also consider:

Leslie Lamport. Specifying concurrent systems with TLA+. Preliminary draft, December 2000. Chapter 8.

K. Mani Chandy and Jayadev Misra. *Parallel Program Design: A Foundation.* Addison-Wesley Publishing Co., Reading, MA, 1988.

E. Allen Emerson and Chin Laung Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 8:275–306, 1987. Earlier version Technical Report CS-TR-84-20, Department of Computer Science, University of Texas, 1984.

Moshe Vardi. Alternating automata: Checking truth and validity for temporal logics. In William McCune, editor, *Automated Deduction-CADE-14 (14th International Conference on Automated Deduction, Townsville, North Queensland, Australia, July, 1997)*, volume 1249 of *Lecture Notes in Computer Science*. Springer Verlag, 1997. `liveness/cade97.ps`.

Paul C. Attie. Liveness-preserving simulation relations. In *ACM Symposium on the Principles of Distributed Computing (PODC)*, Atlanta, Georgia, May 1999. `liveness/Attie-podc99`.

2.2 Tools

Most model-checking tools can be used to prove liveness properties:
[TOOL: SMV]
[TOOL: SPIN]
[TOOL: TLC]

The following thesis describes a tool, built upon Isabelle, for verifying safety and liveness properties of I/O automata:

Olaf Müller. *A Verification Environment for I/O Automata Based on Formalized Meta-Theory.* PhD thesis, Technische Universitat Munchen, 1998.
`liveness/OM-PhD.ps`.

3 Asynchronous Systems with Composition

[4 classes] Leaders: Nancy Lynch, Daniel Jackson, Paul Attie

3.1 Foundations

Composition for I/O automata is described in:

Nancy A. Lynch and Mark R. Tuttle. An introduction to input/output automata. *CWI-Quarterly*, 2(3):219–246, September 1989. Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands. Technical Memo MIT/LCS/TM-373, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, November 1988. `safety/CWI89.ps`.

Nancy Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, March 1996. Chapter 8.

Roberto Segala, Rainer Gawlick, Jørgen Søgaard-Andersen, and Nancy Lynch. Liveness in timed and untimed systems. *Information and Computation*, 141(2):119–171, March 1998. `composition/GLSS-liveness.ps`.

Work on composition using shared variables appears in:

Martin Abadi and Leslie Lamport. Composing specifications. *ACM Transactions on Programming Languages and Systems*, 15(1):73–132, January 1993. Also appeared as SRC Research Report 66. `composition/src66-acm.ps`.

[1] Martin Abadi and Leslie Lamport. Conjoining Specifications. *Toplas*, 17(3):507-534, May, 1995. `composition/src118.ps.Z.`

Leslie Lamport. Composition: A way to make proofs harder. In Willem-Paul de Roever, Hans Langmaack, and Amir Pnueli, editors, *Compositionality: The Significant Difference (Proceedings of the COMPOS'97 Symposium)*, volume 1536 of *Lecture Notes in Computer Science*, pages 402–423, 1998.
`composition/lamport-proofs-harder.pdf`.

Leslie Lamport. Specifying concurrent systems with TLA+. Preliminary draft, December 2000. Chapter 10.

R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999. Earlier version in *Proceedings of the 11th IEEE Symposium on Logic in Computer Science*, pp. 207-218, 1996. `composition/Lics96T.ps`.

Work on the "assume-guarantee" approach to compositional reasoning appears in:

An example of building state machines from other state machines using methods arising from object "inheritance" is described in:

Idit Keidar, Roger Khazan, Nancy Lynch, and Alex Shvartsman. An inheritance-based technique for building simulation proofs incrementally. In *22nd International Conference on Software Engineering (ICSE)*, pages 478–487, Limerick, Ireland, June 2000. `composition/icse00.ps`.

Process algebras, which admit parallel composition and many other kinds of combination operators, are described in:

C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, United Kingdom, 1985.

Robin Milner. *Communicating and mobile systems: the Pi-Calculus.* Cambridge University Press, United Kingdom, 1999. Part I.

Dynamic systems, which include dynamic creation of state machines (which then get composed with the previously-existing state machines) are discussed in:

Robin Milner. *Communicating and mobile systems: the Pi-Calculus.* Cambridge University Press, United Kingdom, 1999. Part II.

Paul Attie and Nancy Lynch. Dynamic Input/Ouput Automata: A Formal Model for Dynamic Computation. Manuscript. `composition/DIOA.ps`.

Paul Attie and Nancy Lynch. Dynamic Input/Output Automata: a Formal Model for Dynamic Systems. Technical Report, College of Computer Science, Northeastern University, January 2001. `composition/Attie-dsioa-tr.ps`.

3.2 Tools

Compositional model-checking is described in:

E. M. Clarke, Orna Grumberg, and Doron Peled. *Model Checking.* MIT Press, December 1999. Chapter 13.

O. Grumberg and D. E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16(3):843–871, May 1994. `papers/composition/Gl-modular.ps`.

The following tool handles reactive modules: [TOOL: Mocha] `http://www.cis.upenn.edu/~mocha/`

Relational models provide ways of describing complex state structure. Some background on models of this kind appears in:

Ian Hayes. *Specification Case Studies.* Prentice Hall, 1993. Second Edition.

Jim Woodcock and Jim Davies. *Using Z: Specification, Refinement, and Proof.* Prentice Hall, 1996.

Martin Fowler. *Analysis Patterns: Reusable Object Models.* Addison Wesley, 1997.

Daniel Jackson. Alloy: A lightweight object modelling notation. Technical Report MIT-LCS-TR-797, MIT Lab for Computer Science, February 2000. To appear in *ACM Transactions on Software Engineering and Methodology.* `composition/alloy-journal.pdf`.

The Alloy analyzer is described in:

Daniel Jackson. Automating first-order relational logic. In *Proceedings of the ACM Conference on Foundations of Software Engineering*, San Diego, CA, November 2000.
`composition/alloy-fse00.pdf`.

[TOOL: The Alloy analyzer]
`http://sdg.lcs.mit.edu/alloy`

4 Timed Systems

[4 classes] Leader: Nancy Lynch

4.1 Foundations

The first batch of sources deals with the timed I/O automaton model:

Nancy Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, March 1996. Chapter 23.

Nancy Lynch and Frits Vaandrager. Forward and backward simulations — Part II: Timing-based systems. *Information and Computation*, 128(1):1–25, July 1996. `timed/IC96.ps`.

Roberto Segala, Rainer Gawlick, Jørgen Søgaard-Andersen, and Nancy Lynch. Liveness in timed and untimed systems. *Information and Computation*, 141(2):119–171, March 1998. `composition/GLSS-liveness.ps`.

Timed automata (without the inputs and outputs) are described in:

Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994. A preliminary version appears in *Proceedings of 17th ICALP, LNCS 443, under the title Automata for modeling real-time systems, pages 322-335, 1990.* `timed/Icalp90.ps`.

Rajeev Alur. Timed automata. NATO summer school, 1998. `timed/Nato97.ps`.

TLA also has ways of dealing with timing. These are discussed in:

Leslie Lamport. Specifying concurrent systems with TLA+. Preliminary draft, December 2000. Chapter 9.

Composition of timed systems

Rajeev Alur and Thomas A. Henzinger. Modularity for timed and hybrid systems. In *Proceedings of Ninth International Conference on Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, 1997. `timed/Concur97.ps`.

We may also briefly visit basic notions of performance modeling/analysis in timed state machine settings.

4.2 Tools

Uppaal is a nice tool for analyzing timed automata; lots of examples have been worked out:
[TOOL: Uppaal] See http://www.docs.uu.se/docs/rtmv/uppaal/

Other work on analyzing timed automata:

S. Yovine. Model checking timed automata. In G. Rozenberg and F. Vaandrager, editors *Lectures on Embedded Systems*, volume 1494 of *Lecture Notes in Computer Science*, October 1998. `timed/Yovine98.ps.gz`.

5 Hybrid Systems

[4 classes] Leader: Nancy Lynch

5.1 Foundations

The following all give general modeling frameworks for hybrid systems:

Nancy Lynch, Roberto Segala, and Frits Vaandraager. Hybrid I/O automata revisited. *Hybrid Systems: Computation and Control: Fourth International Workshop (HSCC'01)*, Lecture Notes in Computer Science, Roma, Italy, March 2001. Springer-Verlag. `hybrid/HIOA.ps`.

Oded Maler, Zohar Manna, and Amir Pnueli. From timed to hybrid systems. In J. W. de Bakker, C. Huizing, and G. Rozenberg, editors, *Real-Time: Theory in Practice (REX Workshop, Mook, The Netherlands, June 1991)*, volume 600 of *Lecture Notes in Computer Science*, pages 447–484. Springer-Verlag, 1992. `hybrid/hybrid-rex.ps`.

R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995. `hybrid/Des94.ps`.

R. Alur, T. A. Henzinger, and P. H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996. `hybrid/Rtss.ps`.

Rajeev Alur and Thomas A. Henzinger. Modularity for timed and hybrid systems. In *Proceedings of Ninth International Conference on Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, 1997. `timed/Concur97.ps`.

Zohar Manna and Henny Sipma. Deductive verification of hybrid systems using STeP. In *Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998. `hybrid/sipma-hscc98.ps`.

Yonit Kesten, Zohar Manna and Amir Pnueli. Verification of Clocked and Hybrid Systems. In *Embedded Systems*, volume 1494 of *Lecture Notes in Computer Science*, Springer-Verlag, 1998. `hybrid/paper-clocked98.ps`.

Yonit Kesten, Zohar Manna and Amir Pnueli. Verifying Clocked Transition Systems. In *Hybrid Systems III*, vol. 1066 of *Lecture Notes in Computer Science*, pages 13-40, Springer-Verlag, 1996. `hybrid/pap-clocks96hybrid3.ps`.

This work has a control-theory flavor. The second paper identifies some continuous methods that can be used with hybrid I/O automata.

Michael S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, June 1995.

Michael S. Branicky, Ekaterina Dolginova, and Nancy Lynch. A toolbox for proving and maintaining hybrid specifications. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems IV (Fourth International Conference on Hybrid Systems, Ithaca, NY, October 1996)*, volume 1273 of *Lecture Notes in Computer Science*, pages 18–30. Springer Verlag, 1997. `hybrid/BDL-toolbox.ps`.

5.2 Tools

The HyTech tool is the best-known tool for modeling and analyzing hybrid systems:

Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HyTech: a model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122, 1997. `hybrid/hytech.ps`.

Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996)*, pages 278–292, 1996. `hybrid/the_theory_of_hybrid_automata.ps`.

Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998. `hybrid/whats_decidable_about_hybrid_automata.ps`.

[TOOL: Hytech] `http://www-cad.eecs.berkeley.edu/~tah/HyTech/`

6 Probabilistic Systems

[2 classes] Leader: Nancy Lynch

6.1 Foundations

A basic model for randomized distributed algorithms and systems appears in:

Anna Pogosyants, Roberto Segala, and Nancy Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distributed Computing*, 13(3):155-186, July 2000. `probability/PSL98-cgi.ps`.

Roberto Segala. Verification of randomized distributed algorithms. Manuscript.

Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 1995. Also, MIT/LCS/TR-676. `probability/TR-676.tar.gz.`

An interesting open question is to define a basic mathematical model that accommodates both hybrid and probabilistic behavior. I don't know of any references on this topic. This seems like a great basic research question.

6.2 Tools

We don't plan to use any particular tools for modeling and analyzing probabilistic systems. The following sample reference describes a tool for probabilistic performance analysis, based on a probabilistic Petri net model.

G. Ciardo, J. K. Muppala, and K. S. Trivedi. SPNP: Stochastic petri net package. In *Proc. of 3rd International Workshop on Petri Nets and Performance Models*, pages 142–150, Kyoto, Japan, December, 1989. `probability/1989SPNP.ps`.