APPENDIX A

A.1 Constructions on Uppaal Automata

This appendix treats several constructions on Uppaal automata used in the verification described in Section 7.5.

Organization of the section

We start with some notational conventions. We explain in Section A.1.2 how multi–way synchronization can be encoded in Uppaal. Then Section A.1.3 describes how reachability properties of open systems in Uppaal (automata) can be reduced to closed systems (networks). Section A.1.4 deals with input–enabling in Uppaal. Finally, Section A.1.5 explains how trace inclusion between two automata can be verified in Uppaal by reducing it to non–reachability of certain error states.

A.1.1 Notational Conventions

Besides the conventions adopted in Section 7.5.1, we find it convenient to use general boolean expressions in guards and invariants, whereas Uppaal only allows conjunctions in these. Therefore, we use $q \xrightarrow{a,g_1 \vee g_2,r} q'$ as an abbreviation for the two transitions $q \xrightarrow{a,g_1,r} q'$ and $q \xrightarrow{a,g_2,r} q'$. The guard $g \implies g'$ stands as an abbreviation for $\neg g \vee g'$. The negation $\neg g$ is an abbreviation for the guard obtained by replacing every $>$ in g by $\leq$, every $\geq$ by $<$, $\geq$ by $<$, $\leq$ by $<$ and every $\wedge$ by $\vee$ and $\vee$ by $\wedge$.

A.1.2 Encoding Multi-way Synchronization in Uppaal

As explained before, Uppaal only provides binary synchronization. We use committed locations and renaming to enforce synchronization between more than two action labels. If we want to have an $a!$-action in A_0 synchronize with n $a?$-actions in $A_1 \ldots A_n$ ($n > 1$), then the idea is as follows. Relabel a in A_i into a fresh label a_i, for all $i \geq 0$. Whenever A_0 performs an $a_0!$ action, an auxiliary automaton 'catches' it and 'distributes' it over the other automata. More precisely, the auxiliary automaton synchronizes on $a_0!$ and enforces – sequentially but without delay nor interruption – synchronization with $a_1? \ldots a_n?$ in $A_1 \ldots A_n$ respectively.

Example A.1.1 Consider the automata in Figure A.1. Being in their initial locations, either A_2 or A_3 takes an $a?$-action to synchronize with the $a!$-action in A_1. Synchronization of the three automata on $a!$, $a?$ and $a?$, can be mimicked by introducing an auxiliary automaton and renaming of actions, as in Figure A.2. Remark that it is also possible to integrate the auxiliary automaton within one of the other automata.

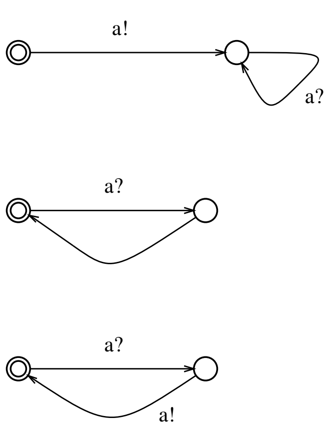

Figure A.1: Three automata: A_1, A_2, A_3

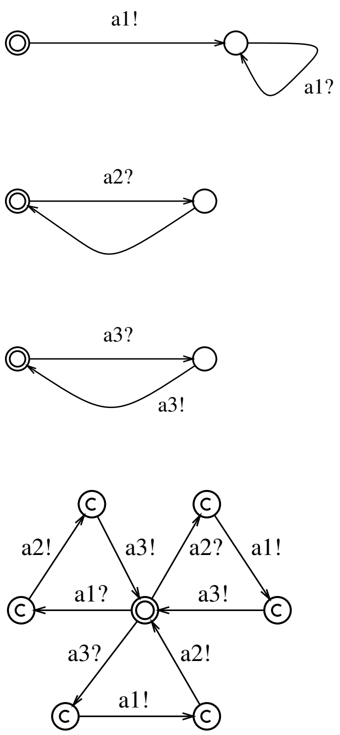

Figure A.2: Encoding multisynchronization with binary synchronization

A.1.3 Reducing Reachability Properties of Automata to Reachability Properties of Networks

This section describes how reachability properties of open systems (automata) can be reduced to closed systems (networks), which can then be verified by Uppaal. The basic idea is simple: we add an auxiliary automaton that synchronizes with every visible action in a given automaton.

Definition A.1.2 Let C be a set of action names. Define the automaton Snc_C as the automaton with one location q_C and transitions $q_C \xrightarrow{a} q_C$ for every a in $C? \cup C!$. For an automaton A, we define $Snc_A = Snc_C$, where C are the action names occurring in A. Moreover, we write q_A for q_C.

The following result allows us to check the reachability properties of an automaton A via the reachability properties the network $\mathcal{N}(A \parallel Snc_A)$, which can be checked by Uppaal. Its proof is straightforward.

Proposition A.1.3 *Let q be a location and v a clock valuation of A. Then (q, v) is reachable in A if and only if $((q, q_A), v)$ is reachable in $(A \parallel Snc_A) \setminus \mathrm{Names}$.*

If A is given in terms of restriction and parallel composition over the components A_1, $A_2, \ldots, A_n$, then we like to carry out the construction of Snc_A on A's components, rather than first computing A explicitly and then performing the construction.

In our verification, A has the form $(A_1 \parallel A_2) \setminus C$. We have checked its reachability properties via the network $\mathcal{N}(A_1, A_2, Snc_{C'})$. This is justified by the following argument, where C is a set of action names, $C' = \mathrm{Names} \setminus C$, $s = (q, v)$ is a state of A and $\overline{s} = ((q, q_{C'}), v)$.

$$
\begin{aligned}
&\overline{s} \text{ reachable in } \mathcal{N}(A_1, A_2, Snc_{C'}) \iff \\
&\overline{s} \text{ reachable in } (A_1 \parallel A_2 \parallel Snc_{C'}) \setminus \mathrm{Names} \iff \\
&\overline{s} \text{ reachable in } (A_1 \parallel A_2 \parallel Snc_{C'}) \setminus C \cup C' \iff \\
&\overline{s} \text{ reachable in } ((A_1 \parallel A_2) \setminus C) \parallel Snc_{C'}) \setminus C' \iff \\
&s \text{ reachable in } (A_1 \parallel A_2) \setminus C.
\end{aligned}
$$

A.1.4 Input Enabling in Uppaal

The *input actions* of an automaton A are the actions of the form $a?$. We call A *input enabled* if synchronization on input actions is always (in any state of the system) possible. More precisely, assume that the $a?$-transitions in A leaving from a location q are given by

$$q \xrightarrow{a?, g_1, r_1} q_1 \ldots, q \xrightarrow{a?, g_n, r_n} q_n.$$

Then A is input enabled iff the expression $\mathrm{Inv}(q) \implies \bigvee_{i=1}^{n}(g_i \wedge \mathrm{Inv}(q_i)[r_i])$ is equivalent to true. Here we use the convention that $\bigvee_{i=1}^{0} \ldots$ yields false. Moreover, $I[r]$ denotes the invariant that is obtained from I by replacing each clock variable in the reset set r by 0. We state that an automaton is input enabled if and only if its underlying TLTS is so, using the standard notion of input enabledness for TLTSs.

A non-input enabled automaton may block input, by being unable to synchronize on it. This is often considered as a bad property, since a component is usually not able to prevent the environment from providing inputs and this might indicate a modeling error. Therefore, it is relevant to know whether blocking of inputs can actually occur in a network of automata. So, one would like to find out whether or not the situation can occur that one component could provide an input to another one, while the latter automaton is not able to synchronize on it. To check this property, we make every component A input enabled by directing every input that would otherwise be blocked to a fresh location, called $\mathit{unexp\_input}_A$. Then we check for reachability of this location. More precisely, we construct the automaton $A^{\mathtt{u\text{-}i}}$ from A as follows. If the outgoing $a?$-transitions leaving from q are as above, then we add the transition $q \xrightarrow{a?, g} \mathit{unexp\_input}_A$, where g is equivalent to $\neg(\mathrm{Inv}(q) \implies \bigvee_{i=1}^{n}(g_i \vee \mathrm{Inv}(q_i)[r_i]))$. (In particular, if q does not have any outgoing $a?$-transitions, then we add $q \xrightarrow{a?} \mathit{unexp\_input}_A$.) We also add the transition $\mathit{unexp\_input}_A \xrightarrow{a?} \mathit{unexp\_input}_A$ for every input action $a?$ of A.

Proposition A.1.4 *The automaton $A^{\mathtt{u\text{-}i}}$ is input enabled.*

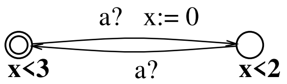

Figure A.3: An input enabled automaton

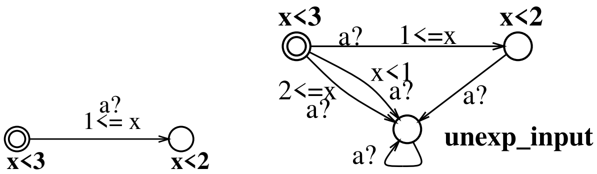

Figure A.4: Constructing an input enabled automaton

Example A.1.5 The automaton in Figure A.3 is input enabled. Notice that $x < 3 \implies ((x < 2)[x := 0])$ yields true. The construction to make an automaton input enabled is shown in Figure A.4. Notice that $\neg(x < 3 \implies 1 \leq x \wedge x < 2)$ is equivalent to $x < 1 \vee x \geq 2$.

By checking for reachability of the location $unexp\_input$, we can establish whether A and $A^{\mathtt{u\_i}}$ are semantically equivalent, i.e. whether their underlying TLTSs are exactly the same when restricted to their reachable states.

Proposition A.1.6 *The network $\mathcal{N}(A_1, \ldots, A_n)$ is semantically equivalent to the network $\mathcal{N}(A_1^{\mathtt{u\_i}}, \ldots, A_n^{\mathtt{u\_i}})$ if and only if none of the locations $unexp\_input_{A_i}$ is reachable in the network $\mathcal{N}(A_1^{\mathtt{u\_i}}, \ldots, A_n^{\mathtt{u\_i}})$.*

A.1.5 Verification of Trace Inclusion

The rest of this section describes how Uppaal can be used in some cases to check whether or not $A \sqsubseteq_{\mathrm{TR}} B$, via the construction of B^{err}, A^f and B^f.

Throughout this section, we assume that the visible actions of A are included in those of B. Recall that we assume that B does not contain committed locations or urgent channels.

The construction of B^{err}

If we want to check whether $A \sqsubseteq_{\mathrm{TR}} B$ for a deterministic automaton B, then we build an automaton B^{err}. This automaton is an adaption of construction in [AD94] and is constructed as follows. First, we add a location *error* to B. Moreover, we add transitions $q \xrightarrow{a\ g} error$ for all locations q and all action labels a such that this transition is enabled if no other a-transition is enabled from q. Finally, we add an internal transition from q to *error* with the guard $\neg\mathrm{Inv}(q)$ and remove all location invariants. This is formalized in the following definition.

Definition A.1.7 The automaton B^{err} is defined as follows.

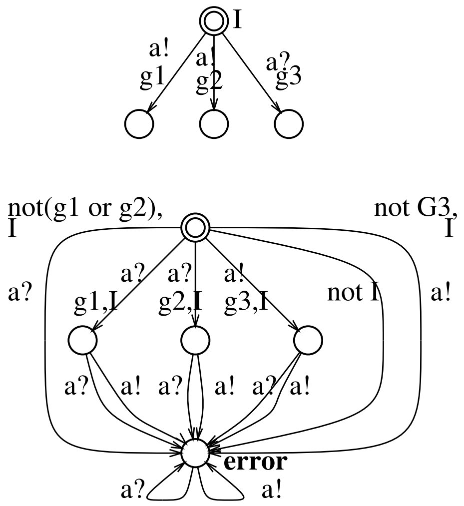

Figure A.5: An automaton and its error-construction

1. The locations of B^{err} are the locations of B together with the (fresh) location *error*,
2. the initial location of B^{err} is the initial location of B,
3. there are no location invariants in B^{err},
4. the transitions of the automaton B^{err} are given as follows, where q and q' range over locations in B, and a over visible actions in B and where $\overline{a}$ toggles the ! and ? in a, thus $\overline{b?} = b!$ and $\overline{b!} = b?$.

$$\begin{aligned}
\bigcup_{q,a,q'} (\{q \xrightarrow{\overline{a},g\wedge I,r}_{B^{err}} q' \mid q \xrightarrow{a,g,r}_{B} q', I = \mathrm{Inv}(q)\} \cup \\
\{q \xrightarrow{\neg I} \mathit{error} \mid I = \mathrm{Inv}(q)\} \cup \\
\{q \xrightarrow{\overline{a},g_{aq}\wedge I} \mathit{error} \mid I = \mathrm{Inv}(q), \\
g_{aq} = \neg \bigvee \{g \mid q \xrightarrow{a,g,r}_{B} q'\}\} \cup \\
\{\mathit{error} \xrightarrow{a} \mathit{error}\}).
\end{aligned}$$

Example A.1.8 Figure A.5 illustrates an automaton and its error-construction.

For a construction of B^{err} in the presence of urgent channels and shared variables, see [Jen99, JLS00]. This work also describes how to encode, what are called, *timed ready simulation relations* as reachability properties.

To decide whether $A \sqsubseteq_{\mathrm{TR}} B$, we check whether the *error*-location is reachable in the network consisting of A and B^{err}. Note that B^{err} is not deterministic, even not if B is so.

Lemma A.1.9 *Let B be a deterministic automaton. Every finite sequence over $actB \cup \mathbb{R}^{>0}$ is a trace of B^{err}. Such a sequence is a trace of B if and only if none of the executions in B^{err} with this trace reach the $error$ location.*

Proposition A.1.10 *Assume that B is a deterministic automaton. Then $A \sqsubseteq_{\mathrm{TR}} B \iff$ error is not reachable in $(A \parallel B^{err}) \setminus \mathrm{Names}$.*

The construction of A^r and B^r

If B is non-deterministic, then we can try to make it deterministic by renaming its labels and then use the above method to verify trace inclusion.

Definition A.1.11 A *renaming function* is a function $h : \mathrm{Names} \to \mathrm{Names} \cup \{\tau\}$. For an automaton (resp. a TLTS) A, we denote by A^h the automaton (resp. the TLTS) obtained by replacing every visible action $a?$ in A by $h(a)?$ and $a!$ by $h(a)!$.

Lemma A.1.12 *Let A, B be automata (resp. TLTSs) and let h be a renaming function on A. Then*

$$A \sqsubseteq_{\mathrm{TR}} B \implies A^h \sqsubseteq_{\mathrm{TR}} B^h.$$

The preceding result shows that, for proving $A_1 \sqsubseteq_{\mathrm{TR}} B_1$, it suffices to find A_2, B_2 and a renaming function h such that: B_2 is deterministic, ${A_2}^h = A_1$, ${B_2}^h = B_1$ and $A_2 \sqsubseteq_{\mathrm{TR}} B_2$. The latter can be verified by Uppaal. Notice that $A_2 \not\sqsubseteq_{\mathrm{TR}} B_2$ does not imply $A_1 \not\sqsubseteq_{\mathrm{TR}} B_1$. Moreover, even if $A_1 \sqsubseteq_{\mathrm{TR}} B_1$, it is not always possible to find the required A_2, B_2 and h. (For instance, if A_1 consists of the transition $s \xrightarrow{a} s'$ and B_1 of $t \xrightarrow{\tau} t'$ and $t' \xrightarrow{a} t''$, then no such a h exists – note that τ is not in the domain of h.)

In our verification, we have constructed h, A_2 and B_2 from A_1, B_1 and a given (conjectured) step refinement f. First, we describe this construction for TLTSs L_1 and K_1, yielding TLTSs L_2 and K_2. Informally, a step refinement from L_1 to K_1 is a function f from the states of L_1 to the states of K_1 such that every transition $s \xrightarrow{a} s'$ of L_1 can be mimicked in K_1 by the transition $f(s) \xrightarrow{a} f(s')$. Here, we also allow internal transitions in L_1 to be mimicked by remaining in the same state in K_1. Formally, a *step refinement* from L_1 to K_1 is a function mapping the state space of L_1 to the state space of K_1 such that, if s is an initial state of L_1, then $f(s)$ is an initial state in K_1. For all transitions $s \xrightarrow{a} t$ of L_1 leaving from a reachable state s, we require that either $f(s) \xrightarrow{a}_{K_1} f(t)$ or that a is internal and $f(s) = f(t)$. A step refinement from an automaton A to an automaton B is a step refinement from the underlying TLTS of A to the underlying TLTS of B.

The idea behind the construction of L_2 and K_2 (on the level of TLTSs) from f and L_1 and K_1 is as follows. The step refinement f yields a correspondence between transitions in K_1 and transitions in L_1: the transition $s' \xrightarrow{a} t'$ in L_1 corresponds to all transitions in K_1 with $s \xrightarrow{a} t$ with $f(s) = s'$ and $f(t) = t'$. To remove the nondeterministic choices in K_1, we rename the actions on corresponding transitions with the same, fresh label, both in K_1 and in L_1. The TLTSs obtained from L_1 and K_1 in this way are denoted by ${L_1}^f$ and ${K_1}^f$ respectively.

Formally, the TLTSs ${L_1}^f$ and ${K_1}^f$ are obtained from L_1 and K_1 as follows: We start with the same state spaces and transition relations as in L_1 and K_1 respectively. Then, for all sources of nondeterminism in K_1, that is, all transitions $t \xrightarrow{a} t'$ in K_1 such that either $a = \tau$ or $\exists t'' \neq t'[t \xrightarrow{a}_{K_1} t'']$, we change the label a into $c_{t,a,t'}$. This yields $t \xrightarrow{c_{t,a,t'}}_{{K_1}^f} t'$ in ${K_1}^f$. Then, for all transitions in L_1, we replace $s \xrightarrow{a} s'$ by $s \xrightarrow{c_{f(s),a,f(s')}}_{{L_1}^f} s'$ in ${L_1}^f$ if and only if $f(s) \xrightarrow{c_{f(s),a,f(s')}}_{{K_1}^f} f(s')$. We require for the relabeling function $c_{\cdot,\cdot,\cdot}$ that

1. the label $c_{s,a,t}$ is not a visible action of A and B,

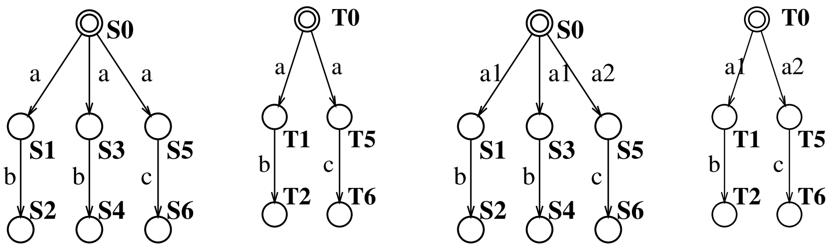

Figure A.6: The TLTSs L_1, K_1 and ${L_1}^f$ and ${K_1}^f$

2. $c_{s,a,t} \neq c_{s,a,t'}$ for $t \neq t'$,
3. $c_{s,a,t} \neq \tau$, and
4. $c_{s,a,s'} \neq c_{t,b,t'}$ for $a \neq b$.

Here, requirement 1 ensures that $c_{s,a,t}$ is fresh. The second and third requirements ensure that outgoing transitions of the same state get different labels and that no actions are relabeled into τ. The last requirement ensures that transitions with different labels get different labels after relabeling. Together these requirements ensure that the relabeled automaton obtained by the construction above is deterministic.

Example A.1.13 Consider the TLTSs in Figure A.6, where we omitted time passage actions $s \xrightarrow{d} s$, where d can be arbitrary. It is clear that the function $s_i \mapsto t_i$ for $i = 0, 1, 2, 5, 6$ and $s_3 \mapsto t_1, s_4 \mapsto t_2$ is a step refinement from L_1 to K_1 and that both the transitions $s_0 \xrightarrow{a} s_1$ and $s_0 \xrightarrow{a} s_3$ should synchronize with the transition $t_0 \xrightarrow{a} t_1$. Hence these transitions get the same action label by the renaming. Similarly, $s_0 \xrightarrow{a} s_1$ and $s_0 \xrightarrow{a} s_3$ should synchronize and get the same labels after renaming (but different from the three labels mentioned previously).

Now, take h to be the identity on the names of A and B and let $h(c_{s,a,t}) = a$ for the new names. Note that τ is not in the domain of h, but that $h(c)$ can be τ. In the example above, h is given by $\{a_1 \mapsto a, a_2 \mapsto a, b \mapsto b, c \mapsto c\}$. Now, we have the following.

Lemma A.1.14 *Let L and K be TLTSs, f a function from the state space of L to the state space of K, and h as defined in the preceding text. Then*

1. ${L^f}^h = L$ *and* ${K^f}^h = K$.
2. $L^f \sqsubseteq_{\mathrm{TR}} K^f \implies L \sqsubseteq_{\mathrm{TR}} K$.

Moreover, we claim that $A^f \sqsubseteq_{\mathrm{TR}} B^f$ if f is a step refinement from A to B. (Then Lemma A.1.14 yields that $A \sqsubseteq_{\mathrm{TR}} B$, which is a basic result for step refinements.) We do not use this claim in the verification.

Since TLTSs are usually infinite, it is relevant to investigate whether this construction on TLTSs can be lifted to automata. This means that, if L_1 is the TLTS underlying A and K_1 the one underlying B, we wish to find automata A^f and B^f whose underlying TLTSs are ${L_1}^f$ and ${L'}^f$ respectively.

We conjecture that this construction can indeed be lifted to the level of automata, provided that r is given by a (finite) expression over the clocks and locations of the automaton. Although the existence of a step refinement already implies trace inclusion, this construction (on automata) would yield a method to check this inclusion with Uppaal. Although a relabeling procedure can be complex in general, in our case study it is much faster than checking whether the function r is a step refinement.

Bibliography

[AAB00] A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th International Conference on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 419–434. Springer-Verlag, 2000.

[ABK98] L. Aceto, A. Burgueño, and K.G.Larsen. Model checking via reachabilty testing for timed automata. In *Proc. of the 3rd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems,* Lisbon, Portugal, volume 1384 of *Lecture Notes in Computer Science*, pages 263–280. Springer-Verlag, 1998.

[ABS01] A. Annichini, A. Bouajjani, and M. Sighireanu. TReX: a tool for reachability analysis of complex systems. In G. Berry, H. Comom, and A. Finkel, editors, *Proceedings of the 13th International Conference on Computer Aided Verification,* Paris, France, volume 2102 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[AD94] R. Alur and D.L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.

[Agg94] S. Aggarwal. Time optimal self-stabilizing spanning tree algorithms. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 1994. Available as Technical Report MIT/LCS/TR-632.

[AHJ01] L. de Alfaro, T.A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In K. G. Larsen and M. Nielsen, editors, *Proceedings CONCUR 01,* Aalborg, Denmark, volume 2154 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[AHV93] R. Alur, T.A. Henzinger, and M.Y. Vardi. Parametric real-time reasoning. *25th Annual ACM Symposium on Theory of Computing (STOC'93)*, pages 592–601, 1993.

[AL91] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991.

[AL92] M. Abadi and L. Lamport. An old-fashioned recipe for real time. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings*

REX Workshop on Real-Time: Theory in Practice, Mook, The Netherlands, June 1991, volume 600 of *Lecture Notes in Computer Science*, pages 1–27. Springer-Verlag, 1992.

[Alf97] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[Alf99] L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In J.C.M. Baeten and S. Mauw, editors, *Proceedings CONCUR 99,* Eindhoven, The Netherlands, volume 1664 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[Alu98] R. Alur. Timed automata. In *NATO-ASI Summer School on Verification of Digital and Hybrid Systems*. Springer-Verlag, 1998. To appear.

[And98] Don Anderson. *FireWire system architecture: IEEE 1394*. MindShare, Inc., 1998. ISBN 0-201-69470-0.

[And99a] S. Andova. Process algebra with interleaving probabilistic parellel composition. Technical Report CSR 99-04, Eindhoven University of Technology, 1999.

[And99b] S. Andova. Process algebra with probabilistic choice. In *Proceedings of 5th AMAST Workshop on Real-Time and Probabilistic Systems (ARTS'99)* Bamberg, Germany, May 1999, volume 1601 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[Apt81] K.R. Apt. Ten years of Hoare's logic: A survey–part I. *ACM Transactions on Programming Languages and Systems*, 3(4):431–483, 1981.

[BA95] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513. Springer-Verlag, 1995.

[Bai96] C. Baier. Polynomial time algorithms for testing probabilistic bisimulation and simulation. In R. Alur and T.A. Henzinger, editors, *Proceedings of the 8th International Conference on Computer Aided Verification,* New Brunswick, NJ, USA, volume 1102 of *Lecture Notes in Computer Science*, pages 38–49. Springer-Verlag, July/August 1996.

[Bai98] C. Baier. On algorithmic verification methods for probabilistic systems. Habilitation Thesis, Universität Mannheim, 1998.

[BBK87] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. On the consistency of Koomen's fair abstraction rule. *Theoretical Computer Science*, 51(1/2):129–176, 1987.

[BBS95] J. C. M. Baeten, J. A. Bergstra, and S. A. Smolka. Axiomatizing probabilistic processes: ACP with generative probabilities. *Information and Computation*, 121(2):234–255, 1995.

[BDG97] M. Bernardo, L. Donatiello, and R. Gorrieri. A tutorial on EMPA: a theory of concurrrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 1997.

[BDM+98] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A Model-Checking Tool for Real-Time Systems. In Hu and Vardi [HV98], pages 546–550.

[BEM00] C. Baier, B. Engelen, and M. Majster–Cederbaum. Deciding bisimilarity and similarity for probabilistic systems. *Journal of Computer and System Sciences*, 60(1):187–231, 2000.

[BH97] C. Baier and H. Hermanns. Weak bisimulation for fully probabilistic processes. In O. Grumberg, editor, *Proceedings of the 9th International Conference on Computer Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 119–130. Springer-Verlag, 1997.

[BJ91] K.G. Larsen B. Jonsson. Specification and refinement of probabilistic processes. In *Proceedings* 6^{th} *Annual Symposium on Logic in Computer Science,* Amsterdam, pages 266–277. IEEE Computer Society Press, 1991.

[BK86] J.A. Bergstra and J.W. Klop. Verification of an alternating bit protocol by means of process algebra. In W. Bibel and K.P. Jantke, editors, *Math. Methods of Spec. and Synthesis of Software Systems '85, Math. Research 31*, pages 9–23, Berlin, 1986. Akademie-Verlag. First appeared as: Report CS-R8404, CWI, Amsterdam, 1984.

[BK90] J.C.M. Baeten and J.W. Klop, editors. *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[BK98] C. Baier and M.Z. Kwiatkowska. On the verification of qualitative properties of probabilistic processes under fairness constraints. *Information Processing Letters*, 66(2):71–79, 1998.

[BLdRT00] G. Bandini, R.L. Lutje Spelberg, R.C.H. de Rooij, and W.J. Toetenel. Application of parametric model checking – the root contention protocol using LPMC. In *Proceedings of the 7th ASCI Conference,* Beekbergen, The Netherlands, Febuary 2000, pages 73–85, 2000.

[BRJ99] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley, Reading, Massachusetts, USA, 1999.

[BS87] T. Bolognesi and S.A. Smolka. Fundamental results for the verification of observational equivalence: a survey. In H. Rudin and C.H. West, editors, *Protocol Specification, Testing, and Verification, VII,* Zürich, Switzerland, pages 165–178. North-Holland, May 1987.

[BS00] C. Baier and M.I.A. Stoelinga. Norm functions for probabilistic bisimulations with delays. In J. Tiuryn, editor, *Proceedings of 3rd International Conference on Foundations of Science and Computation Structures (FOSSACS),* Berlin, Germany, March 2000, volume 1784 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2000.

[BST00] G. Bandini, R.L. Lutje Spelberg, and W.J. Toetenel. Parametric verification of the IEEE 1394a root contention protocol using LPMC. In *Proceedings of the 7th International Conference on Real-Time Computing Systems and Applications (RTCSA 2000),* Cheja Island, South Korea, December 2000, 2000.

[Cas93] C.G. Cassandras. *Discrete event systems. Modeling and Performance Analysis*. Aksen Associates – Irwin, 1993.

[CC92] L. Christoff and I. Christoff. Efficient algorithms for verification of equivalences for probabilistic processes. In K.G. Larsen and A. Skou, editors, *Proceedings of the 3rd International Workshop on Computer Aided Verification,* Aalborg, Denmark, volume 575 of *Lecture Notes in Computer Science*, pages 310–321. Springer-Verlag, 1992.

[CDSY99] R. Cleaveland, Z. Dayar, S. A. Smolka, and S. Yuen. Testing preorders for probabilistic processes. *Information and Computation*, 154(2):93–148, 1999.

[CGP99] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, Cambridge, Massachusetts, 1999.

[CHM90] J. Cheriyan, T. Hagerup, and K. Mehlhorn. Can a maximum flow be computed in $\o(nm)$ time? In M. Paterson, editor, *Proceedings* 17^{th} *ICALP,* Warwick, volume 443 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[Chr90a] I. Christoff. Testing equivalence and fully abstract models of probabilistic processes. In Baeten and Klop [BK90].

[Chr90b] I. Christoff. *Testing equivalences for probabilistic processes*. PhD thesis, Department of Computer Science, Uppsala University, Sweden, 1990.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. McGraw-Hill, Inc., 1990.

[Coh80] D.L. Cohn. *Measure Theory*. Birkhaeuser, Boston, 1980.

[Coh89] L.J. Cohen. *An introduction to the philosophy of induction and probability*. Clarendon Press, Oxford, 1989.

[CS01] A. Collomb–Annichini and M. Sighireanu. Parameterized reachability analysis of the IEEE 1394 Root Contention Protocol using TReX. In P. Pettersson and S. Yovine, editors, *Proceedings of the Workshop on Real-Time Tools (RT-TOOLS'2001)*, 2001.

[CW87] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the* 19^{th} *Annual ACM Symposium on Theory of Computing,* New York City, pages 1–6. ACM, 1987.

[D'A99] P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, University of Twente, November 1999. Available via `http://wwwhome.cs.utwente.nl/~dargenio`.

[dAKN+00] L. de Alfaro, M. Z. Kwiatkowska, G. Norman, D. Parker, and R. Segala. Symbolic model checking of concurrent probabilistic processes using MTBDDs and the Kronecker representation. In S. Graf and M. Schwarzbach, editors, *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems,* Berlin, Germany, volume 1785 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[DGJP99] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labeled Markov systems. In *International Conference on Concurrency Theory*, pages 258–273, 1999. full version available via `http://www-acaps.cs.mcgill.ca/~ prakash`.

[DGRV00] M.C.A. Devillers, W.O.D. Griffioen, J.M.T Romijn, and F.W. Vaandrager. Verification of a leader election protocol: Formal methods applied to IEEE 1394. *Formal Methods in System Design*, 16(3):307–320, June 2000.

[DHKK01] P.R. D'Argenio, H.H. Hermans, J.-P. Katoen, and R. Klaren. Modest - a modelling and description language for stochastic timed systems. In *Proceeding of PAPM–ProbMiV 2001,* Aachen, Germany, volume 2165 of *Lecture Notes in Computer Science*, pages 87–104. Springer-Verlag, 2001.

[Dij69] E. W. Dijkstra. Structured programming. In *Second NATO Conference on Software Engineering Techniques, Rome, Italy*, pages 84–88. NATO, 1969.

[Dil98] D. Dill. Timing assumptions and verification of finite-state concurrent systems. In Hu and Vardi [HV98], pages 197–212.

[DKRT97a] P.R. D'Argenio, J.-P. Katoen, T.C. Ruys, and J. Tretmans. The bounded retransmission protocol must be on time! In E. Brinksma, editor, *Proceedings of the Third Workshop on Tools and Algorithms for the Construction and Analysis of Systems,* Enschede, The Netherlands, volume 1217 of *Lecture Notes in Computer Science*, pages 416–431. Springer-Verlag, April 1997.

[DKRT97b] P.R. D'Argenio, J.-P. Katoen, T.C. Ruys, and J. Tretmans. The bounded retransmission protocol must be on time! In *Proc. of the 3rd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems,* Enschede, The Netherlands, volume 1217 of *Lecture Notes in Computer Science*, pages 416–431. Springer-Verlag, 1997.

[DNH84] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[DP01] J. Desharnais and P. Panangaden. Continuous stochastic logic characterises bisimulation of continuous–time Markov processes, 2001. available via `http://www-acaps.cs.mcgill.ca/ ~ prakash/`.

[Fer90] J.-C. Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of Computer Programming*, 13:219–236, 1989/1990.

[FLP85] M.J. Fischer, N.A. Lynch, and M.S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.

[FS01] C. Fidge and C. Shankland. But what if i don't want to wait forever? In J.M.T. Romijn S. Maharaj, C. Shankland, editor, *Proceedings of the International Workshop on Application of Formal Methods to the IEEE1394 Standard* Berlin, March 2001, 2001. submitted.

[GBS94] R. Gupta, S. Bhaskar, and S.A. Smolka. On randomization in sequential and distributed algorithms. *ACM Computing Surveys*, 28 (1):7–86, 1994.

[gDUoT] TVS group Delft University of Technology. PMC: Prototype Model Checker. `http://tvs.twi.tudelft.nl/toolset.html`.

[GHR93] N. Götz, U. Herzog, and M. Rettelbach. TIPP - a stochastic process algebra. In *Proceedings of the Workshop on Process Algebra and Performance Modelling*, 1993. Technical Report,University of Edinburgh.

[GJS90] A. Giacalone, C. C. Jou, and S. A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In M. Broy and C.B. Jones, editors, *Proceedings of the Working Conference on Programming Concepts and Methods*, pages 443–458. North–Holland, 1990.

[Gla93] R.J. van Glabbeek. The linear time – branching time spectrum II (the semantics of sequential systems with silent moves). In E. Best, editor, *Proceedings CONCUR 93,* Hildesheim, Germany, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer-Verlag, 1993.

[Gla01] R.J. van Glabbeek. The linear time — branching time spectrum I. The semantics of concrete, sequential processes. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 3–99. North-Holland, 2001.

[GLV97] S.J. Garland, N.A. Lynch, and M. Vaziri. IOA: A language for specifiying, programming, and validating distributed systems, September 1997. Available through URL `http://larch.lcs.mit.edu:8001/~garland/ioaLanguage.html`.

[Gly89] P. W. Glyn. 1989. *Proceedings of IEEE*, 77 (1):4–23, 1989.

[GSST95] R.J. van Glabbeek, S.A. Smolka, B. Steffen, and C.M.N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In *Proceedings* 5^{th} *Annual Symposium on Logic in Computer Science,* Philadelphia, USA, pages 130–141. IEEE Computer Society Press, 1995.

[GV98] W.O.D. Griffioen and F.W. Vaandrager. Normed simulations. In A.J. Hu and M.Y. Vardi, editors, *Proceedings of the 10th International Conference on Computer Aided Verification,* Vancouver, BC, Canada, volume 1427 of *Lecture Notes in Computer Science*, pages 332–344. Springer-Verlag, 1998.

[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. North-Holland, 1989. Full version available as Report CS-R9120, CWI, Amsterdam, 1991.

[Hac75] I. Hacking. *The emerge of Probability*. Cambridge University Press, New York, 1975.

[Hal50] P.R. Halmos. *Measure Theory*. Van Nostrand Reinhold Company Inc, New York, 1950.

[Han91] H.A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. PhD thesis, Department of Computer Systems, Uppsala University, 1991. DoCS 91/27.

[Han94] H.A. Hansson. *Time and Probability in Formal Design of Distributed Systems*, volume 1 of *Real-Time Safety Critical Systems*. Elsevier, 1994.

[Har87] D. Harel. Statecharts: a visual approach to complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.

[Her99] H.H. Hermanns. *Interactive Markov Chains*. PhD thesis, University of Erlangen–Nürnberg, July 1999. Available via `http://wwwhome.cs.utwente.nl/~hermanns`.

[HHK95] M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proceedings 36th IEEE Symposium on Foundations of Computer Science*, pages 453–462. IEEE, 1995.

[HHW97] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. In *Software Tools for Technology Transfer*, volume 1, 1997. Also available via `http://www-cad.eecs.berkeley.edu/ ~tah/-HyTech/`.

[Hil96] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[HJ90] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Proc. IEEE Real-Time Systems,* Orlando, Florida*, 1990*, pages 278–287. IEEE Computer Society Press, 1990.

[HJ94] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):515–535, 1994.

[HJL93] C. Heitmeyer, R. Jeffords, and B. Labaw. A benchmark for comparing different approaches for specifying and verifying real-time systems. In *Proceedings 10th International Workshop on Real-Time Operating Systems and Software*, 1993.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, Englewood Cliffs, 1985.

[HP00] O.M. Herescu and C. Palamidessi. Probabilistic asynchronous π-calculus. In J. Tiuryn, editor, *Proceedings of 3rd International Conference on Foundations of Science and Computation Structures (FOSSACS),* Berlin, Germany, March 2000, volume 1784 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2000.

[HRSV01] T.S. Hune, J.M.T. Romijn, M.I.A. Stoelinga, and F.W. Vaandrager. Linear parametric model checking of timed automata. In T. Margeria and W. Yi, editors, *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems,* Genova, Italy, volume 2031 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[HRSV02] T.S. Hune, J.M.T. Romijn, M.I.A. Stoelinga, and F.W. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 2002. To appear.

[HSLL97] K. Havelund, A. Skou, K. G. Larsen, and K. Lund. Formal modelling and analysis of an audio/video protocol: An industrial case study using uppaal. In *Proc. of the 18th IEEE Real-Time Systems Symposium,* San Francisco CA, USA, pages 2–13, 1997.

[HT92] T. Huynh and L. Tian. On some equivalence relations for probabilistic processes. *Fundamenta Informaticae*, 17:211–234, 1992.

[Hui01] M. Huisman. *Reasoning about JAVA programs in higher order logic with PVS and Isabelle*. PhD thesis, Computer Science Institute, February 2001.

[Huy50] Chr. Huygens. *Van Rekeningh in spelen van geluck (in Dutch)*. 1650. Latin version *De rationiniis in ludo aleae* appeared in 1660. A modern dutch version, annotated and translated by W. Kleijne has been published by *epsilon uitgaven*, Utrecht.

[HV98] A.J. Hu and M.Y. Vardi, editors. *Proceedings of the 10th International Conference on Computer Aided Verification,* Vancouver, BC, Canada, volume 1427 of *Lecture Notes in Computer Science*. Springer-Verlag, June/July 1998.

[IEE96] IEEE Computer Society. IEEE Standard for a High Performance Serial Bus. Std 1394-1995, August 1996.

[IEE00a] IEEE Computer Society. IEEE Standard for a High Performance Serial Bus – Amendment 1. Std 1394a-2000, May 2000.

[IEE00b] IEEE Computer Society. P1394.3 Draft Standard for a High Performance Serial Bus (Supplement). Draft 1.2, June 2000.

[IEE01a] IEEE Computer Society. P1394.3 Draft Standard for a High Performance Serial Bus (Supplement). Draft 0.17, June 2001.

[IEE01b] IEEE Computer Society. P1394b Draft Standard for a High Performance Serial Bus (Supplement). Draft 1.11, March 2001.

[Jen99] H.E. Jensen. *Abstraction-Based Verification of Distributed Systems*. PhD thesis, Department of Computer Science, Aalborg University, Denmark, June 1999.

[JLS00] H. E. Jensen, K.G. Larsen, and A. Skou. Scaling up Uppaal - automatic verification of real-time systems using compositionality and abstraction. In J. Mathai, editor, *Proceedings of the 6th International School and Symposium on Formal Techniques and Fault Tolerant Systems (FTRTFT00),* Pune, India, September 2000, volume 1926 of *Lecture Notes in Computer Science*, pages 19–30. Springer-Verlag, 2000.

[JS90] C-C. Jou and S.A. Smolka. Equivalences, congruences and complete axiomatizations for probabilistic processes. In Baeten and Klop [BK90], pages 367–383.

[JY01] B. Jonsson and W. Yi. Compositional testing preorders for probabilistic processes. *Theoretical Computer Science*, 2001.

[KHR97] L. Kühne, J. Hooman, and W.P. de Roever. Towards mechanical verification of parts of the IEEE P1394 serial bus. In I. Lovrek, editor, *Proceedings of the 2nd International Workshop on Applied Formal Methods in System Design,* Zagreb, pages 73–85. University of Zagreb, Faculty of Electrical Engineering and Computing, 1997.

[KL92] A. Skou K. Larsen. Compositional verification of probabilistic processes. In W.R. Cleaveland, editor, *Proceedings CONCUR 92,* Stony Brook, NY, USA, volume 630 of *Lecture Notes in Computer Science*, pages 456–471. Springer-Verlag, 1992.

[KLL+97] K.J. Kristoffersen, F. Laroussinie, K.G. Larsen, P. Pettersson, and W. Yi. A compositional proof of a real-time mutual exclusion protocol. In M. Bidoit and M. Dauchet, editors, *Proceedings TAPSOFT'97: Theory and Practice of Software Development,* Lille, France, volume 1214 of *Lecture Notes in Computer Science*, pages 565–579. Springer-Verlag, April 1997.

[KNP02] M. Z. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In P. Stevens and J.P. Katoen, editors, *Proceedings of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems,* Grenoble, France, Lecture Notes in Computer Science. Springer-Verlag, April 2002. to appear.

[KNS01] M. Z. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of deadline properties in the ieee1394 firewire root contention protocol. In J.M.T. Romijn S. Maharaj, C. Shankland, editor, *Proceedings of the International Workshop on Application of Formal Methods to the IEEE1394 Standard* Berlin, March 2001, 2001. submitted.

[KNSS01] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 268, 2001.

[KS90] P. Kannelakis and S. Smolka. CCS expressions, finite state processes and three problems of equivalence. *Information and Computation*, pages 43–68, 1990.

[LaF97] D. LaFollette. SubPHY Root Contention, Overhead transparencies, August 1997. Available through URL `ftp://gatekeeper.dec.com/pub/-standards/io/1394/ P1394a/Documents/97-043r0.pdf`.

[Lam87] L. Lamport. A fast mutual exclusion algorithm. *ACM Transactions on Computer Systems*, 5(1):1–11, February 1987.

[LLPY97] K.G. Larsen, F. Larsson, P. Pettersson, and Wang Yi. Efficient verification of real-time systems: Compact data structures and state-space reduction. In *Proceedings of the 18th IEEE Real-Time Systems Symposium*, pages 14–24. IEEE Computer Society press, 1997.

[LPY97] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Springer International Journal of Software Tools for Technology Transfer*, 1(1+2):134–152, 1997. Also available via `http://www.docs.uu.se/docs/rtmv/-papers/ lpw-sttt97.pdf`.

[LR81] D. Lehmann and M. Rabin. On the advantage of free choice: a symmetric and fully distributed solution to the dining philosophers problem. In *Proceedings of the 8th ACM Symposium on Principles of Programming Languages*, pages 133–138, 1981.

[LS91] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.

[LSS94] N.A. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. In *Proceedings of the 13th Annual ACM Symposium on the Principles of Distributed Computing*, pages 314–323, Los Angeles, CA, 1994.

[LT89] N.A. Lynch and M.R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, September 1989.

[LTA98] R.F. Lutje Spelberg, W.J. Toetenel, and M. Ammerlaan. Partition refinement in real-time model checking. In A.P. Ravn and H. Rischel, editors, *Proceedings of the Fifth International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98),* Lyngby, Denmark, volume 1486 of *Lecture Notes in Computer Science*, pages 143–157. Springer-Verlag, 1998.

[Lut97] S.P. Luttik. Description and formal specification of the Link layer of P1394. In I. Lovrek, editor, *Proceedings of the 2nd International Workshop on Applied Formal Methods in System Design,* Zagreb, pages 43–56, 1997. Also available as Report SEN-R9706, CWI, Amsterdam. See URL `http://www.cwi.nl/~luttik/`.

[LV95] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations, I: Untimed systems. *Information and Computation*, 121(2):214–233, 1995.

[LV96a] N.A. Lynch and F.W. Vaandrager. Action transducers and timed automata. *Formal Aspects of Computing*, 8(5):499–538, 1996.

[LV96b] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations, II: Timing-based systems. *Information and Computation*, 128(1):1–25, July 1996.

[Lyn96] N.A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., San Fransisco, California, 1996.

[MCB84] M. Ajmone Marsan, G. Conte, and G. Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2(2):93–122, 1984.

[McI99] A.K. McIver. Quantitative program logic and performance. In J.-P. Katoen, editor, *Proceedings of 5th AMAST Workshop on Real-Time and Probabilistic Systems (ARTS'99)* Bamberg, Germany, May 1999, volume 1601 of *Lecture Notes in Computer Science*, pages 19–32. Springer-Verlag, 1999.

[McM92] K.L. McMillan. The SMV system, draft, February 1992. Available through URL `www.cs.cmu.edu/~modelcheck/smv.html`.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall International, Englewood Cliffs, 1989.

[MM99] C.C. Morgan and A.K. McIver. pGCL: formal reasoning for random algorithms. In *South African Computer Journal*, volume 22, pages 14–27, 1999. Special issue on the 1998 Winter School on Formal and Applied Computer Science.

[MMT91] M. Merritt, F. Modugno, and M. Tuttle. Time constrained automata. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91,* Amsterdam, volume 527 of *Lecture Notes in Computer Science*, pages 408–423. Springer-Verlag, 1991.

[MS00] S. Maharaj and C. Shankland. A survey of formal methods applied to leader election in IEEE 1394. *Journal of Universal Computer Science*, pages 1145–1163, 2000.

[NCI99] M. Narashima, R. Cleaveland, and P. Iyer. Probabilistic temporal logics via the modal mu-calculus. In W. Thomas, editor, *Proceedings of the International Conference on Foundations of Science and Computation Structures (FOSSACS)*, volume 1578, pages 288–305, 1999.

[Nyu97] Takayuki Nyu. Modified Tree-ID Process for Long-haul Transmission and Long PHY_DELAY, Overhead transparencies, 1997. Available through URL `ftp://gatekeeper.dec.com/pub/standards/-io/1394/P1394a/Documents/97-051r1.pdf`.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, 5^{th} *GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[Pet96] I. Peterson. *Fatal Defect, chasing killer computer bugs*. Vintage, 1996.

[PSL97] A. Pogosyants, R. Segala, and N.A. Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. In M. Mavronicolas and Ph. Tsigas, editors, *Proceedings of 11th International Workshop on Distributed Algorithms (WDAG'97),* Saarbrucken, Germany, September 1997, volume 1320 of *Lecture Notes in Computer Science*, pages 111–125. Springer-Verlag, 1997. Also, Technical Memo MIT/LCS/TM-555, Laboratory for Computer Science, Massachusetts Institute of Technology.

[PSL98] A. Philippou, O. Sokolsky, and I. Lee. Weak bisimulation for probabilistic systems. In D. Sangiorgi and Robert de Simone, editors, *Proceedings CONCUR 98,* Nice, France, volume 1466 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[PT87] R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987.

[Put94] M. Puterman. *Markov Decision Processes*. John Wiley and Sons, 1994.

[RE98] W.P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison*. Cambridge Tracts in Theoretical Computer Science 47. Cambridge University Press, 1998.

[Rei85] W. Reisig. *Petri nets – an introduction*. EATCS Monographs on Theoretical Computer Science, Volume 4. Springer-Verlag, 1985.

[SC01] R. Segala and S. Cattani. Personal communication, 2001.

[Seg95a] R. Segala. Compositional trace–based semantics for probabilistic automata. In *Proc. CONCUR'95*, volume 962 of *Lecture Notes in Computer Science*, pages 234–248, 1995.

[Seg95b] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1995. Available as Technical Report MIT/LCS/TR-676.

[Seg96] R. Segala. Testing probabilistic automata. In *Proc. CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*, pages 299–314, 1996.

[SGSL98] R. Segala, R. Gawlick, J.F. Søgaard-Andersen, and N.A. Lynch. Liveness in timed and untimed systems. *Information and Computation*, 141(2):119–171, March 1998.

[Sha99] C. Shankland. Using E-LOTOS to pick a leader. *Proceedings of the Workshop on Formal Methods Computation,* Ullapool, UK, September 1999, pages 143–162, 1999.

[SL95] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[SS01] D.P.L. Simons and M.I.A. Stoelinga. Mechanical verification of the IEEE 1394a root contention protocol using Uppaal2k. *Springer International Journal of Software Tools for Technology Transfer (STTT)*, 2001. Accepted for publication.

[Sto99a] M.I.A. Stoelinga. Gambling for leadership: Verification of root contention in IEEE 1394. Technical Report CSI-R9904, Computing Science Institute, University of Nijmegen, 1999.

[Sto99b] M.I.A. Stoelinga. Gambling for leadership: Verification of root contention in IEEE 1394. Technical Report CSI-R9904, Computing Science Institute, University of Nijmegen, 1999.

[Sto01] M.I.A. Stoelinga. Fun with FireWire: Experiences with verifying the IEEE1394 Root Contention Protocol. In J.M.T. Romijn S. Maharaj, C. Shankland, editor, *Proceedings of the International Workshop on Application of Formal Methods to the IEEE1394 Standard* Berlin, March 2001, pages 35–38, 2001. Also, Technical Rapport CSI-R0107, Computing Science Institute, University of Nijmegen, March 2001. Submitted.

[SV99a] C. Shankland and A. Verdejo. Time, E-LOTOS and the FireWire. In M. Ajmone Marsan, J. Quemada, T. Robles, and M. Silva, editors, *Proceedings of the Workshop on Formal Methods and Telecommunications, (WFMT'99)* Zaragoza, Spain, September 1999, pages 103–119. Univ. of Zaragoza Press, 1999.

[SV99b] M.I.A. Stoelinga and F.W. Vaandrager. Root Contention in IEEE 1394. In J.-P. Katoen, editor, *Proceedings of 5th AMAST Workshop on Real-Time and Probabilistic Systems (ARTS'99)* Bamberg, Germany, May 1999, volume 1601 of *Lecture Notes in Computer Science*, pages 53–75. Springer-Verlag, 1999. Also, Technical Rapport CSI-R9905, Computing Science Institute, University of Nijmegen, May 1999.

[SV02a] M.I.A. Stoelinga and F.W. Vaandrager. Gambling together in Monte Carlo: Step refinements for probabilistic automata. Technical Report CSI-R02xx, Computing Science Institute, University of Nijmegen, 2002. In preparation.

[SV02b] M.I.A. Stoelinga and F.W. Vaandrager. Testing scenario's for probabilistic automata. Technical Report CSI-R0201, Computing Science Institute, University of Nijmegen, 2002. To appear.

[Tan81] A.S. Tanenbaum. *Computer networks*. Prentice-Hall International, Englewood Cliffs, 1981.

[Upp] Uppaal distribution. Available via `http://www.uppaal.com`.

[Var85] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *IEEE Symposium on Foundations of Computer Science*, pages 327–338, 1985.

[Wei89] W.P. Weijland. *Synchrony and asynchrony in process algebra*. PhD thesis, University of Amsterdam, 1989.

[Wol98] P. Wolper. Verification: dreams and reality. Inaugural lecture to the course *the algorithmic verification of reactive systems*, 1998. available via the URL `www.montefiore.ulg.ac.be/~pw/cours/francqui.html`.

[WSS97] S.-H. Wu, S.A. Smolka, and E. Stark. Composition and and behaviors of probabilistic I/O automata. *Theoretical Computer Science*, 1-2:1–38, 1997.

[Yi90] Wang Yi. Real-time behaviour of asynchronous agents. In Baeten and Klop [BK90], pages 502–520.

[Yi94] W. Yi. Algebraic reasoning for real-time probabilistic processes with uncertain information. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *Proceedings of the Third International School and Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'94),* Lübeck, Germany, September 1994, volume 863 of *Lecture Notes in Computer Science*, pages 680–693. Springer-Verlag, 1994.

[YL92] W. Yi and K. Larsen. Testing probabilistic and nondeterministic processes. *Proc. Protocol, Specification, Testing, Verification XII*, pages 47–61, 1992.

[Yov98] S. Yovine. Model checking timed automata. In G. Rozenberg and F.W. Vaandrager, editors, *Lectures on Embedded Systems*, volume 1494 of *Lecture Notes in Computer Science*, pages 114–152. Springer-Verlag, October 1998.