Using Lossy Channel Systems for the Verification of Communication Protocols

Parosh Aziz Abdulla and Mats Kindahl
Uppsala University
Dept. of Computer Systems
P.O. Box 325, 751 05 Uppsala, Sweden
E–mail:{parosh,matkin}@docs.uu.se *

Abstract

We consider the class of finite-state systems communicating through unbounded but lossy FIFO channels (called *lossy channel systems*). These systems have infinite state spaces due to the unboundedness of the channels. Lossy channel systems are able to model e.g. link protocols such as the Alternating Bit Protocol, HDLC, and the Bounded Retransmission Protocol. In an earlier paper, we showed that the problems of checking reachability, safety properties, and eventuality properties are decidable for lossy channel systems. The decidability results were obtained under the assumption that message delivery through a channel may take an arbitrary amount of time. In this paper, we consider a variant of lossy channel systems where message deliveries have zero delays. This new model allows us to analyze new classes of protocols which are not covered by the original model.

1 Introduction

In the last few years, there has been a considerable interest in algorithmic verification of distributed and parallel systems. The research has led to the discovery of numerous efficient methods for the verification of *finite-state* systems ([9], [14], [23], [34], etc.). An obvious limitation of these methods is that systems with infinitely many states fall beyond their capabilities. Recently, algorithmic verification methods have been developed for some classes of infinite-state systems, such as certain types of real-time systems that operate on clocks [7, 16, 28], data-independent systems [25, 35], systems with many identical processes [15, 19, 32], context-free processes ([10, 13, 12]), and Petri nets ([24]). In an earlier work ([1]), we considered the class of finite-state systems that communicate via unbounded but lossy FIFO channels (called *lossy channel systems*), and proved that several interesting verification problems are decidable for these systems. More precisely, we showed the decidability of checking:

- *Reachability*: is a set of given states of such a system reachable from some other state of the system.
- *Safety properties*: do all computations of a system stay within a *regular set* of allowed finite traces. This problem can be verified via a transformation to the reachability problem.
- *Eventuality properties*: do all computations of a system eventually reach a given set of states. This result was also shown independently by Finkel [18].

These results were quite surprising in view of the fact that all nontrivial verification problems are undecidable for the class of finite-state systems that communicate via unbounded *perfect* FIFO channels (e.g. [8]).

The decidability results were obtained under the assumption that a message could reside inside a channel for an arbitrary amount of time. In this paper, we consider a variant of lossy channel systems, where messages are not delayed inside channels. This means that if a message is successfully transmitted through a channel, then it will appear instantly at the receiving end. This model can be used to analyze protocols where the time needed to deliver a message through a channel is negligible compared to computation times of other parts of the system. This allows us e.g. to describe the property that an acknowledgment will never arrive after a time-out has occurred.

In this work, we show the decidability of checking reachability and safety properties for zero-delay lossy channel systems. The decidability is achieved in several steps. First, we modify the model introduced in [1] by adding an operation which checks emptiness of channels. Then we consider three decidability problems, namely checking reachability for (1) zero-delay

*Supported in part by the Swedish Research Council for Engineering Sciences (TFR) under contract No. 92-814.

lossy channel systems; (2) lossy channel systems with the ability of testing channel emptiness; and (3) lossy channel systems without the ability of testing channel emptiness. We show that (1) is reducible to (2), and (2) is reducible to (3). The decidabilty of (3) was shown in [1]. As an illustration we give an example of a bounded retransmission protocol [22]. Notice that the reduction from (2) to (3) does not hold e.g. for Petri nets, where zero-testing gives a model as powerful as multi-counter machines which in turn can simulate Turing machines.

Related Work Considerable attention has been paid to the problem of analyzing systems that communicate over perfect unbounded FIFO channels. All interesting verification problems for these systems are in general undecidable, since the channels may be used to simulate the tape of a Turing Machine [8]. Decidability results have been obtained for limited subclasses. Most problems are decidable if the channel alphabets are of size one (in which case the system may be simulated by Petri Nets [26, 31]), or if the language of each channel is bounded (in which case the system becomes finite-state [21, 11]).

Algorithms for partial verification, which may or may not succeed in analyzing a given system, have been developed by Purushotaman and Peng [30] and by Brand and Joyner [8]. These works do not characterize a class of systems for which their method works. Finkel [17] presents a limited class of systems for which verification is decidable; this class does not cover e.g. the Alternating Bit protocol. Sistla and Zuck [33] present a verification procedure for reasoning about a certain set of temporal properties over systems with FIFO channels. The method is not powerful enough to reason about arbitrary finite state machines.

Pachl [29] shows that the reachability problem is decidable if the set of reachable states of the system for each control state consists of a set of channel contents that constitute a recognizable language. It can be proven that this property holds for any system with lossy FIFO channels. In this way, one obtains an alternative proof of decidability for the reachability problem.

Wolper [35] shows that by using an assumption of data-independence, the problem of proving that a data-independent system satisfies the specification of a perfect FIFO channel can be transformed into a verification problem for finite-state systems. This result is different from ours and the above, since we prove properties *about* a system with FIFO buffers.

In [1, 3] we showed the decidability of checking reachability, safety properties, and eventuality properties for lossy channel systems. In [2, 4] we showed that most of the interesting verification problems that were not proven decidable in [1] are in fact undecidable. More precisely, we proved the undecidability of (i) model checking in propositional temporal logics such as CTL and PTL, and (ii) checking eventuality properties under fairness conditions on the channels. In [5] we studied the decidability of several simulation and bisimulation relations between lossy channel systems and finite transition systems. In [6] we showed the above decidability results for the class of *well-structured systems*. This class covers e.g. lossy channel systems, Petri nets, relational automata, and real-time automata.

Outline In Section 2 we introduce a model for lossy channel systems. In Section 3 we study a generalization of the model where we allow testing emptiness of channels. In Section 4 we describe zero-delay lossy channel systems. In Section 5 we present safety properties. In Section 6 we apply the results to a bounded retransmission protocol.

2 Systems with Lossy Channels

In this section, we present the basic definitions of finite-state systems with unbounded but lossy FIFO channels. Intuitively, such a system has two parts: a control part and a channel part. The channel part consists of a set of channels, each of which contains a sequence of messages from a finite alphabet. The control part is a finite-state labeled transition system. Typically, the finite-state part models the total behavior of a number of processes that communicate over the channels. With each transition of the control part there may be associated either some observable interaction with the environment of the system, or an operation on the channels. This channel operation may remove a message from the head of a channel or insert a message at the end of a channel. In addition, a channel can nondeterministically lose messages at any time.

For a set M we use M^* to denote the set of finite strings of elements in M. The empty string is denoted by ε. For sets C and M, a *string vector from C to M* is a function $C \mapsto M^*$. For a string vector w from C to M we use $w[c := x]$ for the string vector w' such that $w'(c) = x$, and $w'(d) = w(d)$, for $d \neq c$. The string vector which maps all elements in C to the empty string is denoted ε.

Definition 2.1 A *Lossy Channel System* $\mathcal{L}$ is a tuple $\langle S, s_0, A, C, M, \delta \rangle$, where

S is a finite set of *control states*,

$s_0 \in S$ is an *initial control state*,

A is a finite set of *actions*,

C is a finite set of *channels*,

M is a finite set of *messages*,

δ is a finite set of *transitions*, each of which is a triple of the form $\langle s_1, op, s_2 \rangle$, where s_1 and s_2 are control states, and op is a label of one of the forms

- $c!m$, where $c \in C$ and $m \in M$,
- $c?m$, where $c \in C$ and $m \in M$,
- a, where $a \in A \cup \{\tau\}$.

Intuitively, the finite-state control part of the lossy channel system $\langle S, s_0, A, C, M, \delta \rangle$ is an ordinary labeled transition system with states S, initial state s_0, and transitions δ. The channel part is represented by the set C of channels, each of which may contain a string of messages in M. The set A denotes a set of observable interactions with the environment. Each transition in δ may either perform an observable interaction in A, the unobservable action τ, or an operation, where

- a transition of the form $\langle s_1, c!m, s_2 \rangle$ represents a change of the control state from s_1 to s_2 while appending the message m to the end of channel c, and where
- a transition of the form $\langle s_1, c?m, s_2 \rangle$ represents a change of the control state from s_1 to s_2 while removing the message m from the head of channel c.

The operational behavior of a lossy channel system is defined by formalizing the intuitive behavior of the system as a labeled transition system with infinitely many states. Let $\mathcal{L}$ be the lossy channel system $\langle S, s_0, A, C, M, \delta \rangle$. A *global state* γ of $\mathcal{L}$ is a pair $\langle s, w \rangle$, where $s \in S$ and w is a string vector from C to M. The *initial global state* γ_0 of $\mathcal{L}$ is the pair $\langle s_0, \varepsilon \rangle$. We shall define a relation $\longrightarrow$ as a set of triples $\langle \gamma, a, \gamma' \rangle$, where γ and γ' are global states, and $a \in A \cup \{\tau\}$. We let $\gamma \xrightarrow{a} \gamma'$ denote $\langle \gamma, a, \gamma' \rangle \in \longrightarrow$. We define $\longrightarrow$ to be the smallest set such that

1. if $\langle s_1, c!m, s_2 \rangle \in \delta$, then $\langle s_1, w \rangle \xrightarrow{\tau} \langle s_2, w[c := w(c) \bullet m] \rangle$, i.e., the control state changes from s_1 to s_2 and m is appended to the end of channel c,
2. if $\langle s_1, c?m, s_2 \rangle \in \delta$, then $\langle s_1, w[c := m \bullet w(c)] \rangle \xrightarrow{\tau} \langle s_2, w \rangle$, i.e., the control state is changed from s_1 to s_2 and m is removed from the head of channel c,
3. if $w(c) = x \bullet m \bullet y$, then $\langle s, w \rangle \xrightarrow{\tau} \langle s, w[c := x \bullet y] \rangle$, i.e., the message m is lost from the contents of channel c without changing the control state, and
4. if $\langle s_1, a, s_2 \rangle \in \delta$, then $\langle s_1, w \rangle \xrightarrow{a} \langle s_2, w \rangle$, i.e., the control state is changed from s_1 to s_2 while the action a is performed.

For global states γ and γ', and a sequence $\sigma \in A^*$, we write $\gamma \overset{\sigma}{\Longrightarrow} \gamma'$ to denote that there is a finite sequence

$$\gamma = \gamma_1 \xrightarrow{a_1} \gamma_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} \gamma_n = \gamma'$$

where σ is the sequence of non-τ actions among $a_1, \ldots, a_{n-1}$. We use $\gamma \longrightarrow \gamma'$ to denote that $\gamma \xrightarrow{a} \gamma'$, for some $a \in A \cup \{\tau\}$, and $\gamma \xrightarrow{*} \gamma'$ to denote that there is a σ such that $\gamma \overset{\sigma}{\Longrightarrow} \gamma'$. A control state s is said to be *reachable from a global state* γ if $\gamma \xrightarrow{*} \gamma'$, where γ' is of the form $\langle s, w \rangle$. A control state s is said to be *reachable* if s is reachable from the initial global state γ_0.

A *computation* is a finite sequence $\gamma_1\ \gamma_2\ \cdots \gamma_n$ of global states, where γ_1 is equal to the initial global state γ_0 and $\gamma_i \longrightarrow \gamma_{i+1}$.

The *reachability problem* for lossy channel systems is the following.

Instance: A lossy channel system $\mathcal{L}$, and a control state s of $\mathcal{L}$.

Question: Is s reachable in $\mathcal{L}$?

Theorem 2.2 *The reachability problem is decidable.*

Proof. The proof can be found in [1]. $\Box$

3 Testing Empty Channels

We modify the model of lossy channel systems to allow the possibility of testing emptiness of channels. A *lossy channel system with emptiness tests* is of the same form as that described in Definition 2.1, except that the label op may have the additional form $\langle\langle c \rangle\rangle$ for some $c \in C$. Intuitively a transition labeled with $\langle\langle c \rangle\rangle$ may be performed only if channel c is empty. The definition of the transition relation $\longrightarrow$ is also modified such that $\langle s_1, w \rangle \xrightarrow{\tau} \langle s_2, w \rangle$, if $\langle s_1, \langle\langle c \rangle\rangle, s_2 \rangle \in \delta$ and $w(c) = \varepsilon$.

Theorem 3.1 *The reachability problem for lossy channel systems with emptiness tests is reducible to the reachability problem for lossy channel systems without emptiness tests.*

Proof. Suppose that we are given a lossy channel system $\mathcal{L} = \langle S, s_0, A, C, M, \delta \rangle$ with emptiness tests, and a control state s in $\mathcal{L}$. We construct a lossy channel system $\mathcal{L}' = \langle S', s_0, A, C, M', \delta' \rangle$ without emptiness tests such that γ is reachable in $\mathcal{L}$ if and only if

s is reachable in $\mathcal{L}'$. We define $M' = M \cup \{\$\}$, where $\$ \notin M$. The idea of the proof is that we simulate checking emptiness of a channel c by first sending \$ to c and then receiving a message from c. It is clear that c is empty if and only if the received message is \$. Formally we take $S' = S \cup \{s_t;\ t \in \delta\}$, i.e. we add a unique new state s_t for each transition t in $\mathcal{L}$. Furthermore we replace each transition t of the from $\langle s_1, \langle\langle c \rangle\rangle, s_2\rangle$ by two transitions, $\langle s_1, c!\$, s_t\rangle$ and $\langle s_t, c?\$, s_2\rangle$.

Now we show that there is a computation in $\mathcal{L}$ reaching s if and only if there is a computation in $\mathcal{L}'$ reaching s. Suppose that there is a computation $\pi = \gamma_1\gamma_2\cdots\gamma_n$ in $\mathcal{L}$ where γ_i is of the form $\langle s_i, w_i\rangle$ and $s_n = s$. We derive a computation π' in $\mathcal{L}'$ from π as follows. If γ_{i+1} is derived from γ_i through a transition of the form $\langle s_i, \langle\langle c \rangle\rangle, s_{i+1}\rangle$, then we replace γ_{i+1} in π by the sequence $\gamma_i'\gamma_{i+1}$ where γ_i' is of the form $\langle s_t, w_i'\rangle$ and $w_i' = w_i[c := \$]$. Otherwise, we copy γ_{i+1} from π to π'. It is clear that π' is a computation in $\mathcal{L}'$ and that the last element of π' is γ_n.

Suppose that there is a computation $\pi' = \gamma_1\gamma_2\cdots\gamma_n$ in $\mathcal{L}'$ where γ_i is of the form $\langle s_i, w_i\rangle$ and $s_n = s$. Observe that it cannot be the case for any i and $t \in \delta$, that $s_i = s_t$ and $s_{i+1} = s_t$, since that would imply that \$ is sent to a channel and then lost immediately, and hence the compuation would deadlock without ever reaching s. We derive a computation π in $\mathcal{L}$ from π' as follows. If γ_i is of the form $\langle s_t, w\rangle$, for some $t \in \delta$, then we know that $w_i(c) = \$$ for some $c \in C$ and that t is of the form $\langle s_i, \langle\langle c \rangle\rangle, s_{i+2}\rangle$. We replace the sequence $\gamma_i\gamma_{i+1}\gamma_{i+2}$ in π' by the sequence $\gamma_i\gamma_{i+2}$ in π. Otherwise, we copy γ_{i+1} from π' to π. It is clear that π is a computation in $\mathcal{L}$ and that the last element of π is γ_n. □

4 Zero-delay Lossy Channel Systems

In this section we consider lossy channel systems where the transmission time of messages is assumed to be negligible. Under this assumption we can formulate restrictions on the order of execution of transitions. We can e.g. demand that a certain transition is executed only if a certain channel is empty. A typical example of such a condition is when we require that a timeout occurs only if there are no messages available to be received from a given channel. Under the zero-delay assumption this amounts to the channel being empty.

A *zero-delay lossy channel system* is of the same form as that described in Definition 2.1, except that the label *op* may have the additional form $\langle\langle c, a\rangle\rangle$ for some $c \in C$ and $a \in A$. Intuitively a transition labelled with $\langle\langle c, a\rangle\rangle$ may be performed only if channel c is empty. The definition of the transition relation $\longrightarrow$ is also modified such that $\langle s_1, w\rangle \xrightarrow{a} \langle s_2, w\rangle$, if $\langle s_1, \langle\langle c, a\rangle\rangle, s_2\rangle \in \delta$ and $w(c) = \varepsilon$.

Theorem 4.1 *The reachability problem for zero-delay lossy channel systems is reducible to the reachability problem for lossy channel systems with emptiness tests.*

Proof. For each transition t of the form $\langle s_1, \langle\langle c, a\rangle\rangle, s_2\rangle$ we create a new control state s_t. We replace t by the two transitions $\langle s_1, \langle\langle c \rangle\rangle, s_t\rangle$ and $\langle s_t, a, s_2\rangle$. □

Theorem 4.2 *The reachability problem is decidable for zero-delay lossy channel systems.*

Proof. The proof follows immediately from Theorem 2.2, Theorem 3.1, and Theorem 4.1. □

5 Safety properties

The reachability problem is related to so-called safety properties. An intuitive characterization of safety properties is that "nothing bad will ever happen". A *trace* of a lossy channel system (of any kind) $\mathcal{L}$ is a sequence $\sigma \in A^*$ such that $\gamma_0 \stackrel{\sigma}{\Longrightarrow} \gamma$ for some γ. We denote the set of traces of $\mathcal{L}$ by $Traces(\mathcal{L})$. Letting Σ denote the set of acceptable sequences of observable actions, safety properties of traces of $\mathcal{L}$ can be formulated as follows.

Instance: A (zero-delay) lossy channel system $\mathcal{L} = \langle S, s_0, A, C, M, \delta\rangle$ and a set $\Sigma \subseteq A^*$ of strings over A.

Question: Does $Traces(\mathcal{L}) \subseteq \Sigma$ hold?

A positive answer to the question means that the system satisfies the property represented by Σ.

If Σ is a regular set then there is a procedure for transforming the problem of deciding safety properties into the problem of deciding reachability [34, 20]. The transformation proceeds as follows.

1. Construct a finite automaton $\mathcal{M}$ that accepts the complement of Σ.

2. Form the product of $\mathcal{L}$ and $\mathcal{M}$ in which $\mathcal{L}$ and $\mathcal{M}$ synchronize over transitions with actions in A.

3. The problem of deciding whether $\mathcal{L}$ satisfies the safety property represented by Σ has now been transformed to the question whether a state of the product in which the $\mathcal{M}$-component is accepting is reachable.

6 A Bounded Retransmission Protocol

In this section we consider a variant of the bounded retransmission protocol (BRP) described in [22]. We model the protocol as a zero-delay lossy channel system. The protocol serves for the transmission of large data packets over unreliable channels. It splits each data packet and transmits it in parts. In case of failure of transmission, only a limited number of retries are allowed (bounded retransmission), hence the protocol may give up the delivery of a part of the packet. Both the sending and the receiving client are informed adequately.

The protocol consists of a *Sender* equipped with a timer T_1, and a *Receiver* equipped with a timer T_2. The sender and the receiver exchange data via two lossy channels, K and L.

The input is read on the "input" port and divided into a number of *blocks*. The blocks are transmitted one by one through channel K. Ideally, each block is delivered on the "output" port. Each block is accompanied by small amount of overhead indicating an alternating bit and whether the block is the first one, the last one, or neither. Here, we assume that each input is divided into at least three blocks.

Sender The sender reads an input message and divides it into a number of blocks (at least three) and then starts sending the blocks one by one over K to the receiver. Timer T_1 is set and a frame is sent into channel K. This frame consists of an overhead (described above) and a datum. Comparing to the model of Section 2, each message represents a block, thus we can define $M = \{0, 1, 2, 3, 4, 5\}$, where the messages $0, 2, 4$ have an alternating bit equal to 0, messages $1, 3, 5$ have an alternating bit equal to 1, messages $0, 1$ are first blocks, messages $4, 5$ are last blocks and messages $2, 3$ are blocks which are neither first nor last. After having sent the frame, the sender waits for an acknowledgement from the receiver, or for a timeout. In case an acknowledgement arrives, the timer T_1 is reset and (depending on whether this was the last block) the sending client is informed of correct transmission (described by an event S_{OK}), or the next block is sent. In case the last acknowledgement fails to come, there is no way to know whether the last block has been delivered or not. This is indicated to the sender client by an event S_{DK}.

In this example we assume that a frame may be resent an arbitrary number of times, after which T_1 times out and the transmission is broken off. We assume that acknowledgements do not arrive after T_1 has timed out.

Receiver The receiver waits for a first frame to arrive. This frame is delivered at the receiving client, timer T_2 is started and an acknowledgement is sent over L to the sender. Then the receiver simply waits for more frames to arrive. The receiver remembers whether the previous frame was the last block and the expected value of the alternating bit. Each frame is acknowledged, but it is handed over to the receiving client only if the alternating bit indicates that it is new. In this case timer T_2 is reset. This goes on until T_2 times out. This happens if for a long time no new frame is received, indicating that transmission has been given up. The receiving client is informed, provided the last block has not just been delivered. Upon reception of the last block, the receiver client knows that a successful transmission of the entire input message has been concluded. This is described by an event R_{OK}.

We assume that the sender does not start reading and transmitting the next input message before the receiver has properly reacted to the failure.

Specification A simple safety property which should be satisfied by the protocol is that the sender client is informed of correct transmission only if the receiver client has already been informed. Also, the receiver cannot declare two successful receptions without the sender being informed of the first one. Using a tool for checking safety properties of lossy channel systems [27], we have considered the BRP where input messages are divided into n blocks for different values of n. For each n we have shown that the protocol satisfies the above safety property. The verification times (in seconds) were 3 for $n = 3$, 8 for $n = 4$, and 20 for $n = 6$.

References

[1] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *Proc. 8^{th} IEEE Int. Symp. on Logic in Computer Science*, 1993. Extended Abstract.

[2] P. A. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. In Abiteboul and Shamir, editors, *Proc. ICALP '94*, volume 820 of *Lecture Notes in Computer Science*, pages 316–327. Springer Verlag, 1994. Extended Abstract.

[3] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels, 1995. Accepted for publication in the journal of Information and Computation.

[4] P. A. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels, 1996. Accepted for publication in the journal of Information and Computation.

[5] P. A. Abdulla and M. Kindahl. Decidability of simulation and bisimulation between lossy channel systems and finite state systems. In Lee and Smolka, editors, *Proc. CONCUR '95, 6^{th} Int. Conf. on Concurrency Theory*, volume 962 of *Lecture Notes in*

Computer Science, pages 333 – 347. Springer Verlag, 1995. Extended Abstract.

[6] P. A. Abdulla, K. Čerāns, B. Jonsson, and T. Yih-Kuen. General decidability theorems for infinite-state systems. In *Proc. 11^{th} IEEE Int. Symp. on Logic in Computer Science*, 1996. To appear.

[7] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proc. 5^{th} IEEE Int. Symp. on Logic in Computer Science*, pages 414–425, Philadelphia, 1990.

[8] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 2(5):323–342, April 1983.

[9] J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proc. 5^{th} IEEE Int. Symp. on Logic in Computer Science*, 1990.

[10] O. Burkart and B. Steffen. Model checking for context-free processes. In Cleaveland, editor, *Proc. CONCUR '92, Theories of Concurrency: Unification and Extension*, number 630 in Lecture Notes in Computer Science, pages 123–137. Springer Verlag, 1992.

[11] A. Choquet and A. Finkel. Simulation of linear FIFO nets having a structured set of terminal markings. In *Proc. 8^{th} European Workshop on Applications and Theory of Petri Nets*, 1987.

[12] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation equivalence is decidable for basic parallel processes. In *Proc. CONCUR '93, Theories of Concurrency: Unification and Extension*, pages 143–157, 1993.

[13] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. In W. R. Cleaveland, editor, *Proc. CONCUR '92, Theories of Concurrency: Unification and Extension*, pages 138–147, 1992.

[14] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specification. *ACM Trans. on Programming Languages and Systems*, 8(2):244–263, April 1986.

[15] E. M. Clarke and O. Grumberg. Avoiding the state explosion problem in temporal logic model checking algorithms. In *Proc. 6^{th} ACM Symp. on Principles of Distributed Computing, Vancouver, Canada*, pages 294–303, 1987.

[16] K. Čerāns. Decidability of bisimulation equivalence for parallel timer processes. In *Proc. Workshop on Computer Aided Verification*, volume 663 of *Lecture Notes in Computer Science*, pages 302–315, 1992.

[17] A. Finkel. A new class of analyzable CFSMs with unbounded FIFO channels. In *Protocol Specification, Testing, and Verification VIII*, pages 1–12, Atlantic City, USA, 1988. IFIP WG 6.1, North-Holland.

[18] A. Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7(3), 1994.

[19] S. M. German and A. P. Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.

[20] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. *Formal Methods in System Design*, 2(2):149–164, 1993.

[21] M. Gouda, E. Gurari, T.-H. Lai, and L. Rosier. On deadlock detection in systems of communicating finite state machines. *Computers and Artificial Intelligence*, 6(3):209–228, 1987.

[22] J. Groote and J. van de Pol. A bounded retransmission protocol for large data packets. Technical report, Department of Philosophy, Utrecht University, October 1993.

[23] G. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall, 1991.

[24] P. Jančar. Decidability of a temporal logic problem for petri nets. *Theoretical Computer Science*, 74:71–93, 1990.

[25] B. Jonsson and J. Parrow. Deciding bisimulation equivalences for a class of non-finite-state programs. *Information and Computation*, 107(2):272–302, Dec. 1993.

[26] R. Karp and R. Miller. Parallel program schemata. *Journal of Computer and Systems Sciences*, 3(2):147–195, May 1969.

[27] M. Kindahl. Implementation of a reachability algorithm for systems with unreliable channels. Master's thesis, Dept. of Computer Systems, Uppsala University, Sweden, 1993. Available as report DoCS 93/44.

[28] K. Larsen and W. Yi. Time-abstracted bisimulation: Implicit specification and decidability. In *Mathematical Foundations of Programming Semantics*, Lecture Notes in Computer Science, New Orleans, April 1993. Springer Verlag.

[29] J. Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Protocol Specification, Testing, and Verification VII*, May 1987.

[30] W. Peng and S. Purushothaman. Data flow analysis of communicating finite state machines. *ACM Trans. on Programming Languages and Systems*, 13(3):399–442, July 1991.

[31] L. Rosier and H.-C. Yen. Boundedness, empty channel detection and synchronization for communicating finite automata. *Theoretical Computer Science*, 44:69–105, 1986.

[32] Z. Shtadler and O. Grumberg. Network grammars, communication behaviours and automatic verification. In Sifakis, editor, *Proc. Workshop on Computer Aided Verification*, volume 407 of *Lecture Notes in Computer Science*, pages 151–165. Springer Verlag, 1990.

[33] A. Sistla and L. Zuck. Automatic temporal verification of buffer systems. In Larsen and Skou, editors, *Proc. Workshop on Computer Aided Verification*, volume 575 of *Lecture Notes in Computer Science*. Springer Verlag, 1991.

[34] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1^{st} IEEE Int. Symp. on Logic in Computer Science*, pages 332–344, June 1986.

[35] P. Wolper. Expressing interesting properties of programs in propositional temporal logic (extended abstract). In *Proc. 13^{th} ACM Symp. on Principles of Programming Languages*, pages 184–193, Jan. 1986.