Verifying Programs with Unreliable Channels

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

Abstract

We consider the verification of a particular class of infinite-state systems, namely systems consisting of finite-state processes that communicate via unbounded lossy FIFO channels. This class is able to model e.g. link protocols such as the Alternating Bit Protocol and HDLC. For this class of systems, we show that several interesting verification problems are decidable by giving algorithms for verifying (1) the *reachability problem*: is a finite set of global states reachable from some other global state of the system, (2) *safety properties over traces* formulated as regular sets of allowed finite traces, and (3) *eventuality properties*: do all computations of a system eventually reach a given set of states. We have used the algorithms to verify some idealized sliding-window protocols with reasonable time and space resources. Our results should be contrasted with the well-known fact that these problems are undecidable for systems with unbounded *perfect* FIFO channels.

1 Introduction

During the last decade, the research on methods for algorithmic verification of concurrent and parallel systems has expanded dramatically. Substantial

*Supported in part by the Swedish Board for Industrial and Technical Development (NUTEK) as part of ESPRIT BRA project No. 6021 (REACT), and by the Swedish Research Council for Engineering Sciences (TFR) under contract No. 92-814.

progress has been made in the verification of *finite-state* systems, for which efficient algorithmic verification methods have been developed and successfully applied to e.g. communication protocols and hardware structures ([BCM+90], [CES86], [Hol91], [VW86], etc.). For infinite-state systems, e.g. systems that operate on data from unbounded domains, algorithmic verification is more difficult. In general, verification of infinite-state systems requires a substantial manual effort, since most interesting verification problems are undecidable. 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 [ACD90, Yi91, Č92], data-independent systems [JP93, Wol86], systems with many identical processes [CG87, GS92, SG90], context-free processes ([BS92, CHS92, CHM93]), and Petri nets ([Jan90]). In order to extend the applicability of algorithmic verification, we consider it important to develop analogous techniques also for other classes of infinite-state systems.

A class of systems which has been important in the analysis of e.g. communication protocols consists of finite-state processes that communicate via unbounded FIFO channels [BZ83, Boc78]. Such systems are infinite-state due to the unboundedness of the channels, and it is well-known that most interesting verification problems are undecidable for this class of systems [BZ83]. Several verification methods have been developed for such systems [BZ83, CF87, GGLR87, Pac87, PP91, SZ91], but since the verification problem is undecidable, there is no completely automatic verification method which covers the whole class. In this paper, we consider a variant of this class where the FIFO channels are unreliable, in that they may nondeterministically lose messages. In spite of this restriction, we can model many interesting systems, e.g. link protocols such as the Alternating Bit Protocol [BSW69] and HDLC [ISO79]. These protocols and others are designed to operate correctly even in the case that the FIFO channels are faulty and may lose messages. In order to model and verify such systems, it is therefore sufficient that there is an algorithm for verifying systems that communicate via unbounded but *lossy* FIFO channels.

In this paper, we consider algorithmic verification of finite-state systems that communicate via unbounded but lossy FIFO channels. We show that several interesting verification problems are decidable for such systems. More precisely, we give algorithms for verifying the following classes of properties.

1. The *reachability problem*: is a set of given states of such a system reachable from some other state of the system.

2. *Safety properties:* does a system satisfy a *safety property over traces*, formulated as a regular set of allowed finite traces. This problem can

be verified via a transformation to the reachability problem.

3. A simple class of *eventuality properties*: do all computations of a system eventually reach a given set of states. This result has also been proven independently by Finkel [Fin94]. The class of eventuality properties we consider here correspond to the class of *guarantee properties* in the hierarchy of temporal properties in [MP92]. We make no assumption on fairness in the channels. Thus a system may fail to meet a certain eventuality property because the channels lose all their messages.

Our algorithms show that the above problems are decidable for systems with lossy communication channels. This should be contrasted with the fact the these problems are undecidable for systems with perfect FIFO channels.

The main idea of the algorithm for deciding whether a set N of states is reachable, is to perform a search which analyzes the behavior "backwards" from the set N, trying to find a path to the initial state. Since channels are unbounded, this search is *a priori* unbounded, but two facts make the search bounded. The first fact is that we do not have to analyze a state for which we have already analyzed a "simpler" state. A state is "simpler" than another if the states differ only in that the content of each channel in the first state is a (not necessarily contiguous) substring of the content of the same channel in the second state. The second fact is that by a result in language theory (Higman's theorem) only a finite number of states can be generated if we discard states that have "simpler" variants.

We have presently not determined the complexity of the verification problem. However, some experiments with sliding-window protocols indicate that non-trivial examples can be analyzed with reasonable time and space resources.

An interesting consequence of our result is that our methods and results generalize directly to systems that use other sequence-like data structures that may lose elements. For instance, it follows that for Turing machines with a tape that may nondeterministically lose symbols, properties such as the halting problem are decidable.

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 [BZ83]. 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 [KM69, RY86]), or if the language of each channel is bounded (in which case the system becomes finite-state [GGLR87, CF87]).

Algorithms for partial verification, which may or may not succeed in analyzing a given system, have been developed by Purushotaman and Peng [PP91] and by Brand and Joyner [BZ83]. These works do not characterize a class of systems for which their method works. Finkel [Fin88] 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 [SZ91] 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 [Pac87] 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 [Wol86] 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.

Outline The remainder of the paper is organized as follows. In the next section, we present basic definitions of finite state systems with lossy FIFO channels. In Section 3 we use the definitions to describe the Alternating Bit Protocol. In Section 4 we present the properties that we verify, and describe how to transform arbitrary safety properties to the reachability problem. In Section 5 we present algorithms for deciding these properties, and argue for their correctness. Section 6 contains a few empirical results from running the algorithm. In Section 7 we present conclusions and directions for future research. In the appendix we give proofs for some of the lemmas in the paper.

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 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. For $x, y \in M^*$ we let $x \bullet y$ denote the concatenation of x and y. The empty string is denoted by ε. If $x \neq \varepsilon$, then $first(x)$ ($last(x)$) denotes the first (last) element of x. 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 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 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 $\langle s_1, c?m, s_2 \rangle \in \delta$, and if $w' = w[c := m \bullet w(c)]$, then $\langle s_1, w' \rangle \xrightarrow{\tau} \langle s_2, w \rangle$, i.e., the control state is changed from s_1 to s_2 and m is removed from the head of channel c. Note that if $w(c) = \varepsilon$, or if $first(w(c)) \neq m$, then the transition $\langle s_1, c?m, s_2 \rangle$ cannot be performed from the global state $\langle s_1, w \rangle$.
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.
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 $\gamma \stackrel{\sigma}{\Longrightarrow} \gamma'$ 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 $\gamma \stackrel{\sigma}{\Longrightarrow} \gamma'$. A global state γ' is said to be *reachable* from a global state γ if $\gamma \xrightarrow{*} \gamma'$. A global state γ is said to be *reachable* if γ is reachable from the initial global state γ_0.

3 Example: The Alternating Bit Protocol

In this section we model the well-known Alternating Bit Protocol [BSW69] as a lossy channel system. The alternating bit protocol contains a *Sender* and a *Receiver* that communicate over two FIFO channels c_M (used to transmit messages from the Sender to the Receiver) and c_A (used to transmit acknowledgments from the Receiver to the Sender). Both channels are faulty in the sense that they can lose but not reorder messages.

The purpose of the protocol is to transmit messages from the Sender to the Receiver in correct order, in spite of the fact that the channels can lose messages. Corruption of messages can also be taken into account by modeling it as loss (some mechanism will detect and discard a corrupted message).

The operation of the protocol is the following:

The *Sender* reads a pending message to be sent to the Receiver. It adds a sequence number to the message, sends it over the channel c_M to the Receiver and awaits an acknowledgment from the Receiver with the same sequence number. If it arrives, the procedure is repeated with the next pending message but with sequence numbers inverted. If no acknowledgment arrives within some time period the Sender retransmits the message. Retransmissions are repeated until a corresponding acknowledgment arrives.

The *Receiver* receives messages with accompanying sequence numbers from the channel c_M. When the message has the expected sequence number, the message is delivered, and the Receiver looks for a message with inverted sequence number. Messages with non-expected sequence numbers are discarded. The Receiver sends acknowledgments to the Sender over the channel c_A. An acknowledgment contains the sequence number of the last received message.

In Figure 1 the Sender and the Receiver are represented by labeled transition systems. In our model we have omitted the actual messages, i.e. only sequence numbers are transmitted over the channels. The finite state control part of the lossy channel system is obtained as the combination of these two transition systems. The protocol operates on the two channels c_M and c_A. This means that the model of the Alternating Bit Protocol is the lossy channel system $\langle S, s_0, A, C, M, \delta \rangle$ where

- S is the set of pairs of the form $\langle i, j \rangle$, where $1 \le i, j \le 4$,
- s_0 is the state $\langle 1, 1 \rangle$,
- A is the set $\{Snd, Rcv\}$, where Snd represents the sending of a message by the environment to the protocol, and Rcv represents the reception

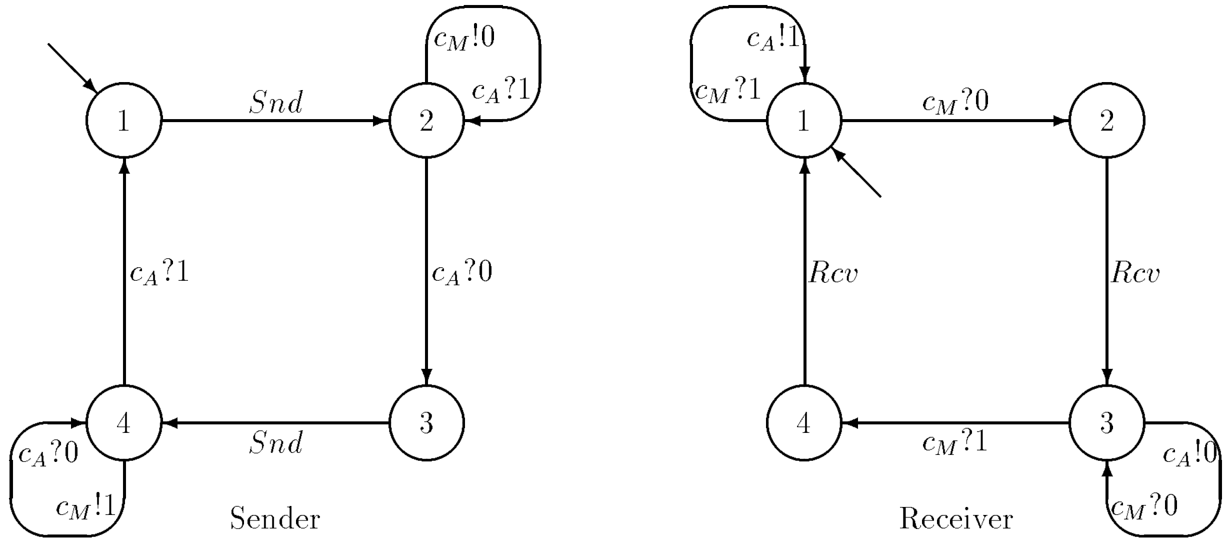

Figure 1: The Sender and the Receiver of the Alternating Bit Protocol

of a message by the environment from the protocol,

C is the set $\{c_M, c_A\}$,

M is the set $\{0, 1\}$, i.e., messages consist of only a sequence number,

δ consists of the tuples of the form $\langle\langle s_1, r_1\rangle, op, \langle s_2, r_2\rangle\rangle$ where either $r_1 = r_2$ and $\langle s_1, op, s_2\rangle$ is a transition in the Sender component or $s_1 = s_2$ and $\langle r_1, op, r_2\rangle$ is a transition in the Receiver component. $\square$

4 Properties of Lossy Channel Systems

In this section, we present the reachability problem, safety properties, and eventuality properties. We also outline a transformation from safety properties to the reachability problem.

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

Instance: A lossy channel system $\mathcal{L}$, and a finite set Γ of global states of $\mathcal{L}$.

Question: Is some state in Γ reachable in $\mathcal{L}$?

Typically, the set Γ may represent states with some undesired property, which we do not want to occur when the system executes. A special case of the reachability problem is whether a certain set of control states is reachable. Formally, a finite set of control states represents an infinite number of global states, but due to the fact that channels may lose messages, it is equivalent to pose the question whether it is possible to reach a control state in the set with all channels empty. This set of global states is finite; note however that an algorithm for deciding the question must consider an infinite state-space of global states.

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". Thus if γ is a "bad" global state, then the property "γ is not reachable" is a safety property.

A class of safety properties can be described by specifying sequences of observable actions in A that are allowed to occur when the system executes. For instance, the property that Snd is the first action, that each Snd action may only be followed by a Rcv action, and that each Rcv action may only be followed by a Snd action can be formulated as the set of sequences

$$(Snd\ \ Rcv)^* \quad \cup \quad (Snd\ \ Rcv)^*\ \ Snd\ .$$

A *trace* of a lossy channel system $\mathcal{L}$ 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 lossy channel system $\mathcal{L} = \langle S, s_0, A, C, \delta, F \rangle$ 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 [VW86, GW93]. 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.

More precisely, we let a finite automaton be a tuple $\langle T, t_0, A, \rho, F\rangle$, where T is a set of states, $t_0 \in T$ is an initial state, A is a set of actions, $\rho \subseteq (T \times A \times T)$ is a transition relation, and $F \subseteq T$ is a set of *accepting states*. The product of a lossy channel system $\mathcal{L} = \langle S, s_0, A, C, M, \delta\rangle$ and a finite automaton $\mathcal{M} = \langle T, t_0, A, \rho, F\rangle$ (note that the sets of actions are the same for the lossy channel system and the finite automaton), denoted $\mathcal{L}\|\mathcal{M}$, is the lossy channel system $\langle S \times T, \langle s_0, t_0\rangle, A, C, M, \delta'\rangle$ where δ' is the set of triples of form $\langle\langle s_1, t_1\rangle, op, \langle s_2, t_2\rangle\rangle$ such that either

- op is of the form $c!m$, $c?m$, or τ, and $t_1 = t_2$, and $\langle s_1, op, s_2\rangle$ is a transition in δ, or
- $op \in A$, $\langle s_1, op, s_2\rangle$ is a transition in δ, and $\langle t_1, op, t_2\rangle$ is a transition in ρ,

The problem of deciding

$$Traces(\mathcal{L}) \subseteq \Sigma$$

has then been transformed into the equivalent problem of deciding

> No state of form $\langle\langle s, t\rangle, \varepsilon\rangle$ where t is an accepting state of $\mathcal{M}$ is reachable in $\mathcal{L}\|\mathcal{M}$.

Example 4.1 The safety property

$$(Snd \;\; Rcv)^* \quad \cup \quad (Snd \;\; Rcv)^* \;\; Snd$$

is represented by the finite automaton in Figure 2 which accepts the complement of the allowed sequences. No two read actions may be performed consecutively, and no two write actions may be performed consecutively. This automaton may be used as a specification for the Alternating Bit protocol in Section 3.

Eventuality Properties In this paper we shall consider a simple class of eventuality properties:

Instance: A lossy channel system $\mathcal{L}$, and a set N of control states of $\mathcal{L}$.

Question: Do all sequences of transitions of $\mathcal{L}$ eventually reach a global state whose control component is in N?

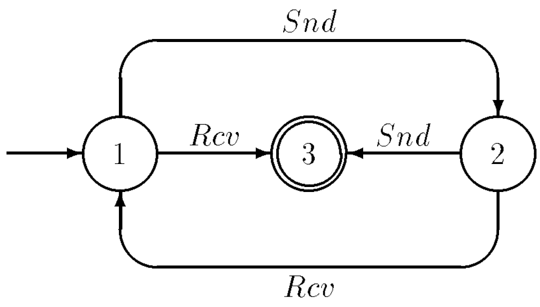

Figure 2: Specification of the Alternating Bit Protocol

5 Algorithms

In this section we give algorithms for deciding the reachability problem and the eventuality problem. Safety properties over traces can be verified from the algorithm for reachability, as described in Section 4.

For $x_1, x_2 \in M^*$, let $x_1 \preceq x_2$ denote that x_1 is a (not necessarily contiguous) substring of x_2. If w_1, w_2 are string vectors from C to M, then $w_1 \preceq w_2$ denotes that $w_1(c) \preceq w_2(c)$ for each $c \in C$. Let $\langle s_1, w_1 \rangle \preceq \langle s_2, w_2 \rangle$ denote that $s_1 = s_2$ and $w_1 \preceq w_2$.

5.1 Deciding the Reachability Problem

The main idea of our algorithm for deciding whether some global state in a set Γ is reachable is to perform a reachability analysis "backwards" from the set Γ, trying to find a path to the initial state. It turns out that it is inconvenient to use the direct inverse of the transition relation $\longrightarrow$, e.g. since this will generate "backwards" paths that add messages to channels in an uncontrolled manner. Instead we define a new "backward" transition relation $\rightsquigarrow$ on global states, which goes in a direction opposite that of $\longrightarrow$ but is not simply the inverse of $\longrightarrow$. One difference is that the $\longrightarrow$-transitions that are caused by message loss in channels are not mirrored in $\rightsquigarrow$. In some cases $\gamma_1 \rightsquigarrow \gamma_2$ denotes that the state γ_1 can be reached from γ_2 by first performing a $\longrightarrow$-transition and thereafter losing a certain message. An important property (described in Theorem 5.3) is that for any global states γ_1 and γ_2, where γ_1 is of the form $\langle s_1, \varepsilon \rangle$, we have $\gamma_1 \stackrel{*}{\longrightarrow} \gamma_2$ if and only if $\gamma_2 \stackrel{*}{\rightsquigarrow} \gamma_1$. This means that we can decide the reachability problem by a backwards reachability analysis from the set Γ. This backward search is not *a priori* bounded. In order to show that the search is finite, we prove in Lemma 5.5 that if $\gamma_1 \preceq \gamma_2$, then for each $\rightsquigarrow$-path from γ_2 to the initial global state there is a shorter or equal-length $\rightsquigarrow$-path from γ_1 to the initial state. This means that we do not need to analyze states for which "simpler" states (with respect to the relation $\preceq$) have been analyzed. Finally, it follows from Higman's theorem (Theorem 5.6) that this fact makes the number of states that must be analyzed finite.

Definition 5.1 Let $\mathcal{L} = \langle S, s_0, A, C, M, \delta \rangle$ be a lossy channel system. Define $\rightsquigarrow$ to be the smallest binary relation on global states such that

1. if $\langle s_2, c!m, s_1 \rangle \in \delta$ then $\langle s_1, w[c := w(c) \bullet m] \rangle \rightsquigarrow \langle s_2, w \rangle$,
2. if $\langle s_2, c!m, s_1 \rangle \in \delta$, $w(c) \neq \varepsilon$, and $last(w(c)) \neq m$, then $\langle s_1, w \rangle \rightsquigarrow \langle s_2, w \rangle$,
3. if $\langle s_2, c!m, s_1 \rangle \in \delta$, and $w(c) = \varepsilon$, then $\langle s_1, w \rangle \rightsquigarrow \langle s_2, w \rangle$,
4. if $\langle s_2, c?m, s_1 \rangle \in \delta$ then $\langle s_1, w \rangle \rightsquigarrow \langle s_2, w[c := m \bullet w(c)] \rangle$,
5. if $\langle s_2, a, s_1 \rangle \in \delta$, then $\langle s_1, w \rangle \rightsquigarrow \langle s_2, w \rangle$. □

In case 2, we could have omitted the condition $last(w(c)) \neq m$. In such a case there would be two $\rightsquigarrow$ -transitions corresponding to $\langle s_2, c!m, s_1 \rangle$ from a global state $\langle s_2, w \rangle$ where $last(w(c)) = m$ (corresponding to case 1 and case 2). However in our algorithm for reachability (which we will describe later in this section) the transition generated by case 2 is always subsumed by the one generated from case 1, and hence we choose to omit it already in the definition of $\rightsquigarrow$.

The relation between $\longrightarrow$ and $\rightsquigarrow$ is captured by the following lemma.

Lemma 5.2 *If γ_1 and γ_2 are global states of a lossy channel system, then*

(a) $(\gamma_1 \rightsquigarrow \gamma_2) \supset \left(\gamma_2 \xrightarrow{*} \gamma_1\right)$

(b) for any global state γ_4 we have

$$\left(\begin{array}{c} \gamma_1 \longrightarrow \gamma_2 \\ \wedge \\ \gamma_4 \preceq \gamma_2 \end{array} \right) \supset \exists \gamma_3. \left(\begin{array}{c} \gamma_3 \preceq \gamma_1 \\ \wedge \\ \gamma_4 \overset{*}{\rightsquigarrow} \gamma_3 \end{array} \right)$$

Proof: (a) Suppose that $\gamma_1 \rightsquigarrow \gamma_2$. Let $\gamma_1 = \langle s_1, w_1 \rangle$ and $\gamma_2 = \langle s_2, w_2 \rangle$. There are five cases to check corresponding to the five cases in the definition of $\rightsquigarrow$.

1. If $\langle s_2, c!m, s_1\rangle \in \delta$ and $w_1 = w_2[c := w_2(c) \bullet m]$. We have $\langle s_2, w_2\rangle \xrightarrow{\tau} \langle s_1, w_1\rangle$.

2. If $\langle s_2, c!m, s_1\rangle \in \delta$, $w_1(c) \neq \varepsilon$, $last(w_1(c)) \neq m$, and $w_1 = w_2$. We have $\langle s_2, w_2\rangle \xrightarrow{\tau} \langle s_1, w_1[c := w_1(c) \bullet m]\rangle$ and $\langle s_1, w_1[c := w_1(c) \bullet m]\rangle \xrightarrow{\tau} \langle s_1, w_1\rangle$.

3. If $\langle s_2, c!m, s_1\rangle \in \delta$, $w_1(c) = \varepsilon$, and $w_1 = w_2$. We have $\langle s_2, w_2\rangle \xrightarrow{\tau} \langle s_1, w_1[c := m]\rangle$ and $\langle s_1, w_1[c := m]\rangle \xrightarrow{\tau} \langle s_1, w_1\rangle$.

4. If $\langle s_2, c?m, s_1\rangle \in \delta$ and $w_2 = w_1[c := m \bullet w_1(c)]$. We have $\langle s_2, w_2\rangle \xrightarrow{\tau} \langle s_1, w_1\rangle$.

5. if $\langle s_2, a, s_1\rangle \in \delta$ and $w_1 = w_2$. We have $\langle s_2, w_2\rangle \xrightarrow{a} \langle s_1, w_1\rangle$.

(b) Suppose that $\gamma_1 \longrightarrow \gamma_2$ and $\gamma_4 \preceq \gamma_2$. Let $\gamma_1 = \langle s_1, w_1\rangle$, $\gamma_2 = \langle s_2, w_2\rangle$, and $\gamma_4 = \langle s_2, w_4\rangle$, where $w_4 \preceq w_2$. There are four cases to check corresponding to the four cases in the definition of $\longrightarrow$. For each case we find w_3 and define γ_3 to be $\langle s_1, w_3\rangle$. In each case it can easily be checked that $\gamma_3 \preceq \gamma_1$ and that $\gamma_4 \overset{*}{\rightsquigarrow} \gamma_3$.

1. If $\langle s_1, c!m, s_2\rangle \in \delta$ and $w_2 = w_1[c := w_1(c) \bullet m]$, then if $last(w_4(c)) = m$ then take w_3 such that $w_4 = w_3[c := w_3(c) \bullet m]$, otherwise if $last(w_4(c)) \neq m$ or $w_4(c) = \varepsilon$ then take $w_3 = w_4$.

2. If $\langle s_1, c?m, s_2\rangle \in \delta$ and $w_1 = w_2[c := m \bullet w_2(c)]$, then take $w_3 = w_4[c := m \bullet w_4(c)]$.

3. If $s_1 = s_2$ and for some x, y we have $w_1(c) = x \bullet m \bullet y$ and $w_2(c) = x \bullet y$, then take $w_3 = w_4$ which makes $\gamma_4 = \gamma_3$.

4. if $\langle s_1, a, s_2\rangle \in \delta$ and $w_2 = w_1$, then take $w_3 = w_4$. □

From Lemma 5.2 we can infer that $\gamma_1 \overset{*}{\rightsquigarrow} \gamma_2$ implies $\gamma_2 \xrightarrow{*} \gamma_1$ (this follows from (a)), and that $\gamma_1 \xrightarrow{*} \gamma_2$ implies that there is a state γ_3 with $\gamma_3 \preceq \gamma_1$ such that $\gamma_2 \overset{*}{\rightsquigarrow} \gamma_3$ (this follows from (b)). In particular, we have the following.

Theorem 5.3 *If $\gamma_1 = \langle s_1, \varepsilon\rangle$ and γ_2 are global states of a lossy channel system, then*

$$\gamma_1 \xrightarrow{*} \gamma_2 \quad \textit{iff} \quad \gamma_2 \overset{*}{\rightsquigarrow} \gamma_1$$

Proof: Follows directly from Lemma 5.2. □

The fact that backward reachabilty of a state can be simulated by backward reachability of a "simpler" state is described by the following lemmas.

Lemma 5.4 *For any global states γ_1, γ_2, and γ_3 of a lossy channel system, we have*

$$\left(\begin{array}{c}\gamma_1 \rightsquigarrow \gamma_2 \\ \wedge \\ \gamma_3 \preceq \gamma_1\end{array}\right) \supset \exists \gamma_4. \left(\begin{array}{c}\gamma_4 \preceq \gamma_2 \\ \wedge \\ \gamma_3 \rightsquigarrow \gamma_4\end{array}\right)$$

Proof: Let $\gamma_1 = \langle s_1, w_1 \rangle$, $\gamma_2 = \langle s_2, w_2 \rangle$, and $\gamma_3 = \langle s_1, w_3 \rangle$, where $w_3 \preceq w_1$. The proof is divided into five cases, corresponding to the cases in the definition of $\rightsquigarrow$. For each case, we describe how to find w_4 and define γ_4 to be $\langle s_2, w_4 \rangle$. In each case it can easily be checked that $\gamma_4 \preceq \gamma_2$, and that $\gamma_3 \rightsquigarrow \gamma_4$.

1. $\langle s_2, c!m, s_1 \rangle \in \delta$ and $w_1 = w_2[c := w_2(c) \bullet m]$. There are three cases:
   (a) If there is a w_3' such that $w_3 = w_3'[c := w_3'(c) \bullet m]$, then take $w_4 = w_3'$.
   (b) If $w_3(c) \neq \varepsilon$ and $last(w_3(c)) \neq m$, take $w_4 = w_3$.
   (c) If $w_3(c) = \varepsilon$, take $w_4 = w_3$.
2. $\langle s_2, c!m, s_1 \rangle \in \delta$, $w_1 = w_2$, $w_1(c) \neq \varepsilon$, and $last(w_1(c)) \neq m$. The proof is similar to that of the previous case.
3. $\langle s_2, c!m, s_1 \rangle \in \delta$, $w_1 = w_2$, and $w_1(c) = \varepsilon$. This means that $w_3(c) = \varepsilon$. Take $w_4 = w_3$.
4. $\langle s_2, c?m, s_1 \rangle \in \delta$, $w_2 = w_1[c := m \bullet w_1(c)]$. Take $w_4 = w_3[c := m \bullet w_3(c)]$.
5. $\langle s_2, a, s_1 \rangle \in \delta$, $w_2 = w_1$. Take $w_4 = w_3$. □

Define the *distance* of a global state γ (denoted $dist(\gamma)$) as the minimal number of $\rightsquigarrow$-transitions needed for coming from γ to the initial state γ_0. If the initial state is not reachable from γ via $\rightsquigarrow$ then define $dist(\gamma) = \infty$.

Lemma 5.5 *If γ_1 and γ_2 are global states of a lossy channel system such that $\gamma_1 \preceq \gamma_2$ then $dist(\gamma_1) \leq dist(\gamma_2)$.*

Proof: Follows directly from Lemma 5.4. □

We are now ready to present the reachability algorithm. The algorithm (displayed in Figure 3) inputs the set Γ of global states, and should check whether Γ is reachable or not. The algorithm maintains a set W, initialized to Γ, of states that have not yet been analyzed, and a set V which contains information about the set of states which have been analyzed. The algorithm preserves the the following invariant: $W \cup V$ is reachable if and only if Γ is reachable, and if Γ is reachable then $(\exists \gamma \in W)\,(\forall \gamma' \in V)\,(dist(\gamma) < dist(\gamma'))$. Thus, if W becomes empty, then the algorithm terminates concluding that Γ is unreachable. Otherwise, the algorithm proceeds by analyzing each state in W in turn. When a state γ in W is analyzed, three possibilities arise:

1. if γ is the initial state then terminate and say that Γ is reachable,
2. if there exists a state γ' in V with $\gamma' \preceq \gamma$, then simply discard γ (since the invariant together with Lemma 5.5 imply that there is a $\gamma'' \in W$ such that $dist(\gamma'') < dist(\gamma') \leq dist(\gamma)$),
3. otherwise generate the $\rightsquigarrow$-successors of γ, put these into W, and move γ from W to V. Furthermore, remove each member γ' of V for which $\gamma \preceq \gamma'$ (By Lemma 5.5 this does not affect the information contained in V).

The correctness of the algorithm follows directly from the invariant. The reason why the algorithm always terminates is that only a finite set of global states can be added to V. This can be explained as follows. Whenever a new element γ is added to V it is ensured that $\gamma' \not\preceq \gamma$, for each γ' already added to V. This means that the sequence of global states added to V forms a sequence $\gamma_1\ \gamma_2\ \gamma_3\ \cdots$, such that $\gamma_i \not\preceq \gamma_j$ for all $i < j$. It follows from Higman's theorem that there is no such sequence which is infinite. This result can be found e.g. as Theorem 6.1.2 in [Lot83], where it is attributed to Higman [Hig52]. A version which suits our purposes is the following.

Theorem 5.6 *(Higman's theorem) Let M be a finite set. There is no infinite sequence $w_1\ w_2\ w_3\ \cdots$ of elements in M^*, such that $w_i \not\preceq w_j$ for all $i < j$.*

It is straightforward to generalize Higman's theorem to sequences $\gamma_1\ \gamma_2\ \gamma_3\ \cdots$ of global states. $\Box$

There is a connection between our algorithm for deciding reachability and standard proofs by invariants. When running the reachability algorithm with an unreachable input set Γ, then upon completion of the algorithm the set V gives a finite characterization of the set of global states from which Γ is reachable. The characterization is described as a set $\hat{V}$, where

$$\hat{V} = \{\gamma;\ \exists \gamma'.\ \gamma' \in V \text{ and } \gamma' \preceq \gamma\}$$

For each global state γ, the set Γ is reachable from γ if and only if $\gamma \in \hat{V}$. If we define the set I to be the complement of $\hat{V}$, then I represents an invariant which can be used to prove that Γ is unreachable. Since Γ is unreachable,

Algorithm
Input: A LCS and a finite set Γ of global states
output: Is Γ reachable?
var W , V : sets of global states
begin
    $W := \Gamma$
    $V := \emptyset$
    while $W \neq \emptyset$ **do**
        if $\gamma_0 \in W$ **then**
            exit(true)
        else
            Let $\gamma \in W$;
            if $(\exists \gamma' \in V)(\gamma' \preceq \gamma)$ **then**
                $W := W - \{\gamma\}$
            else
                $V := \{\gamma\} \cup \{\gamma' \ : \ (\gamma' \in V) \ \wedge \ (\gamma \not\preceq \gamma')\}$
                $W := W \cup \{\gamma' \ : \ \gamma \ \rightsquigarrow \ \gamma'\} - \{\gamma\}$
    od (* while *)
    exit(false)
end

Figure 3: Algorithm for deciding reachability

the initial global state γ_0 is a member of I. Note that I is larger than or equal to the set of the reachable states.

Furthermore, since the channels can lose messages, any invariant in a lossy channel system (I in this case) is closed under the $\preceq$ relation. Using Higman's Theorem it can be shown [Cou91] that any set I which is closed under the $\preceq$ relation is regular and can be represented by a unique finite set V of counter-examples, in the sense that the complement of I is equal to $\hat{V}$.

This fact can be proved as follows. Let I be a set of strings over a finite alphabet, which is closed under the $\preceq$ relation. Let I' be the complement of I. It is clear that I' is closed under the $\succeq$ relation. Let V be the set of minimal elements of I', i.e.

$$V \quad = \quad \{\gamma;\ (\gamma \in I') \ \wedge \ (\not\exists \gamma' \in I'.\ \gamma' \prec \gamma)\}$$

By Higman's theorem it follows that V is finite, and hence the set V consists of a finite number of counter-examples characterizing the set I. This implies that I' (and hence I) is regular.

5.2 Example

Consider the Alternating Bit Protocol, described in Section 3, and the safety property of Example 4.1. Using the method described in Section 4, we can reduce the problem of checking the safety prperty to an instance of the reachability problem, by forming the product of the lossy channel system representing the protocol and the finite automaton (in Figure 2) represnting the complement of the safety property.

The set of the control states of the

6 Empirical Results

As an empirical experiment, we have analyzed some sliding-window protocols [Tan81], using a model where the sender and the receiver communicate via two unbounded and lossy channels; one for transmitting messages from the sender to the receiver and one for transmitting acknowledgments from the receiver to the sender. Each protocol has a parameter $MaxSeq$, where $MaxSeq \geq 2$. The messages and acknowledgments are assigned sequence numbers in the set $\{0, \ldots, MaxSeq - 1\}$. The size of the sender window is $MaxSeq - 1$, while the size of the receiver window is one. In our model we have omitted the actual messages, i.e. only sequence numbers are transmitted

$MaxSeq$	No. of control states	No. of iterations	Size of V	Verification time (secs)
2	48	136	56	0.01
3	216	1049	273	0.1
4	640	4579	856	0.53
5	1500	14408	2100	1.9
6	3024	37883	4404	6.8
7	5488	86559	8281	22
8	9216	179982	14368	64

Table 1: Performance of the algorithm on different sliding window protocols

over the channels. For these protocols, we have verified the safety properties that the traces are included in the traces of a buffer with a capacity of $MaxSeq - 1$.

Notice that if $MaxSeq = 2$, the sliding window protocol described above reduces to the Alternating Bit Protocol (Figure 1), and the specification reduces to that described in Figure 2.

Table 1 illustrates the performance of a draft implementation of the algorithm in the language C for different values of $MaxSeq$. The second column shows the number of control states in the product of the lossy channel system describing the protocol and the finite automaton representing the specification. This number is the product of of the number of states of the Sender ($MaxSeq^2$), the number of states of the Receiver ($2 * MaxSeq$), and the number of states of the specification ($MaxSeq + 1$). The third column shows the number of iterations of the loop when applying the algorithm of Figure 3, the fourth column shows the size of the set V upon termination, and the last column shows the verification time on a Sun SPARCstation 10. As described in [Kin93], the verification time is dependent on the data structures used for the implementation of the sets W and V in the reachability algorithm (see Figure 3). Table 1 describes the results of an implementation where the set W is implemented as a queue and the set V is implemented as a hash table.

7 Conclusion

In this paper, we have shown that several types of safety and liveness properties of systems of finite-state processes that communicate over unbounded but lossy FIFO channels are decidable. We have performed empirical studies that show that the reachability algorithm is practical for verifying idealized models of sliding window protocols of moderate size.

Our results generalize to other types of sequences that can lose elements, e.g. "lossy stacks", "lossy tapes" (of e.g. Turing machines). It follows that the halting problem for Turing machines with "lossy tapes" is decidable.

There is also another way to prove that the reachability problem is decidable. For systems with perfect unbounded FIFO buffers, Pachl [Pac87] has shown that the reachability problem is decidable if the set of reachable global states can be described by combining each control state with a recognizable expression that describes the possible corresponding contents of the channels. We can then combine this fact with a result in language theory which states that any language which is closed under the substring relation ($\preceq$) is recognizable [Cou91], to prove the decidability of the reachability problem for lossy channel systems. Pachl gives no algorithm for constructing a description of the reachable states in the case where the channel contents is a recognizable language. A description can be obtained from our reachability algorithm by inspecting the set of nonreachable states which remain in V when the algorithm terminates, as described in Section 5.

Acknowledgments

We are grateful to Stefan Arnborg, Joachim Parrow, and Wolfgang Thomas for comments and discussions. We thank Wolfgang Thomas for bringing the description of Higman's theorem in [Lot83] to our attention. Special thanks to Ricardo Civalero and Mats Kindahl for assisting with the empirical experiments. The work was supported in part by the Swedish Board for Industrial and Technical Development (NUTEK) as part of ESPRIT BRA project No. 6021 (REACT), and by the Swedish Research Council for Engineering Sciences (TFR) under contract No. 92-814.

References

[ACD90] 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.

[BCM+90] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proc. 5^{th} IEEE Int. Symp. on Logic in Computer Science*, 1990.

[Boc78] G. V. Bochman. Finite state description of communicating protocols. *Computer Networks*, 2:361–371, 1978.

[BS92] 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.

[BSW69] K. Bartlett, R. Scantlebury, and P. Wilkinson. A note on reliable full-duplex transmissions over half duplex lines. *Communications of the ACM*, 2(5):260–261, 1969.

[BZ83] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 2(5):323–342, April 1983.

[Č92] 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.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. 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.

[CF87] 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.

[CG87] 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.

[CHM93] 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.

[CHS92] 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.

[Cou91] B. Courcelle. On constructing obstruction sets of words. *Bulletin of the EATCS*, (44):178–185, June 1991.

[Fin88] 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.

[Fin94] A. Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7(3), 1994.

[GGLR87] M.G. Gouda, E.M. Gurari, T.-H. Lai, and L.E. Rosier. On deadlock detection in systems of communicating finite state machines. *Computers and Artificial Intelligence*, 6(3):209–228, 1987.

[GS92] S. M. German and A. P. Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.

[GW93] 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.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.*, 2:326–336, 1952.

[Hol91] G.J. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall, 1991.

[ISO79] ISO. Data communications – HDLC procedures – elements of procedures. Technical Report ISO 4335, International Standards Organization, Geneva, Switzerland, 1979.

[Jan90] P. Jančar. Decidability of a temporal logic problem for petri nets. *Theoretical Computer Science*, 74:71–93, 1990.

[JP93] 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.

[Kin93] Mats Kindahl. Implementation of a reachability algorithm for systems with unreliable channels. Technical Report 44, Department of Computer Systems, Uppsala University, Dec. 1993.

[KM69] R.M. Karp and R.E. Miller. Parallel program schemata. *Journal of Computer and Systems Sciences*, 3(2):147–195, May 1969.

[Lot83] M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, 1983.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer Verlag, 1992.

[Pac87] J.K. Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Protocol Specification, Testing, and Verification VII*, May 1987.

[PP91] 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.

[RY86] L.E. Rosier and H-C. Yen. Boundedness, empty channel detection and synchronization for communicating finite automata. *Theoretical Computer Science*, 44:69–105, 1986.

[SG90] 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.

[SZ91] A.P. Sistla and L.D. 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.

[Tan81] A. Tanenbaum. *Computer Neworks.* Prentice Hall, 1981.

[VW86] 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.

[Wol86] Pierre 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.

[Yi91] Wang Yi. CCS + Time = an interleaving model for real time systems. In Leach Albert, Monien, and Rodriguez Artalejo, editors, *Proc. ICALP '91*, volume 510 of *Lecture Notes in Computer Science*. Springer Verlag, 1991.