Verifying Programs with Unreliable Channels
(Extended Abstract)

Parosh Abdulla and *Bengt Jonsson*
Dept. of Computer Systems, Uppsala University *

December 8, 1992

Abstract

The research on algorithmic verification methods for concurrent and parallel systems has mostly focussed on finite-state systems, with applications in e.g. communication protocols and hardware systems. For infinite-state systems, e.g. systems that operate on data from unbounded domains, algorithmic verification is more difficult, since most verification problems are in general undecidable. In this paper, 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. The unboundedness of the channels makes these systems infinite-state. For this class of systems, we show that several interesting verification problems are decidable by giving 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. Does a system satisfy an arbitrary *safety property*, formulated as a regular set of allowed finite traces. This problem can be verified via a transformation to the reachability problem.
3. *Eventuality properties*: Do all computations of a system eventually reach a given set of states.

Our results should be contrasted with the well-known fact that these problems are undecidable for systems with unbounded *perfect* FIFO channels.

*Full Address: Uppsala University, Dept. of Computer Systems, Box 325, 751 05 Uppsala, Sweden. Tel: +46 - 18 - 183163 and + 46 - 18 - 183157, Fax: +46 - 18 - 550225, E–mail:parosh@docs.uu.se, bengt@docs.uu.se

1 Introduction

During the last decade, the research on methods for algorithmic verification of concurrent and parallel systems has expanded dramatically. Substantial 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 (e.g. [Hol91]) and hardware structures (e.g. [BCM+90]). 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, in particular since most interesting verification problems are undecidable. Recently, algorithmic verification methods have been developed for some classes of infinite-state systems, e.g. certain types of real-time systems [ACD90] that operate on clocks, and data-independent systems [JP89, Wol86]. 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, is systems consisting 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 they work only in some cases and not in others. By considering certain subclasses of systems with FIFO channels, we may be able to obtain decidability results. In this paper, we consider the subclass 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. Does a system satisfy an arbitrary *safety property*, formulated as a regular set of allowed finite traces. This problem can be verified via a transformation to the reachability problem.
3. *Eventualityg properties*: Do all computations of a system eventually reach a given set of states.

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 is a (not necessarily contiguous) substring of the content of the same channel in the other state. The second fact is that by a result in language theory (Higman's theorem) there is a bound on the number of states that can be generated if we discard states that have "simpler" variants.

We have presently not determined the complexity of the verification problem. We suspect that practical examples will have a much lower complexity than the actual worst-case complexity calculated from the size of the examples. 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 structure 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]). 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. Our algorithm for the reachability problem can be used to construct a representation of this recognizable language.

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 is limited and 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.

Wolper [Wol86] shows that by using an assumption of data-independence, it can be proven that a data-independent system satisfies the specification of perfect FIFO channel. This result is quite 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 present the properties that we verify, and describe how to transform arbitrary safety properties to the reachability problem. In Section 4 we present algorithms for deciding these properties, and argue for their correctness. Section 5 contains a few empirical results from running the algorithm. Finally, in Section 6 we present conclusions and directions for future research. In the full paper, we present an application to a sliding window 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. 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 A we use A^* to denote the set of finite strings of elements in A. For $x, y \in A^*$ we let $x \bullet y$ denote the concatenation of x and y. The empty string is denoted by ε. For sets C and A, a *string vector from C to A* is a function $C \mapsto A^*$. For a string vector w from C to A 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.

Typically, the finite-state part models the total behavior of a number of processes that communicate over the channels in 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* 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* of $\mathcal{L}$ is the pair $\langle s_0, \varepsilon\rangle$. Define $\longrightarrow$ to be the smallest ternary relation on $S \times (A \cup \{\tau\}) \times S$ 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 $\langle s, w\rangle$ and $\langle s', w'\rangle$, and a sequence $\sigma \in M^*$, we write $\langle s, w\rangle \stackrel{\sigma}{\Longrightarrow} \langle s', w'\rangle$ to denote that there is a finite sequence

$$\langle s, w\rangle = \langle s_1, w_1\rangle \xrightarrow{a_1} \langle s_2, w_2\rangle \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} \langle s_n, w_n\rangle = \langle s', w'\rangle$$

where σ is the sequence of non-τ actions among $a_1, \ldots, a_{n-1}$. We use $\langle s, w\rangle \longrightarrow \langle s', w'\rangle$ to denote that $\langle s, w\rangle \xrightarrow{a} \langle s', w'\rangle$, for some $a \in A \cup \{\tau\}$, and $\langle s, w\rangle \xrightarrow{*} \langle s', w'\rangle$ to denote that there is a σ such that $\langle s, w\rangle \stackrel{\sigma}{\Longrightarrow} \langle s', w'\rangle$. A global state $\langle s', w'\rangle$ is said to be *reachable* from a global state $\langle s, w\rangle$ if $\langle s, w\rangle \xrightarrow{*} \langle s', w'\rangle$. A global state $\langle s, w\rangle$ is said to be *reachable* if $\langle s, w\rangle$ is reachable from the initial global state $\langle s_0, \varepsilon\rangle$.

3 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* for lossy channel systems is the following.

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

Question: Is any state in N reachable in $\mathcal{L}$?

Typically, the set N 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.

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 $\langle s, w\rangle$ is a "bad" global state, then the property "$\langle s, w\rangle$ is not reachable" is a safety property.

A class of safety properties can be described by stating 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

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

Letting Σ denote the set of acceptable sequences of observable actions, the problem of deciding safety properties of lossy channel systems is more generally of the following form.

Instance: A 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 $\langle s_0, \varepsilon\rangle \stackrel{\sigma}{\Longrightarrow} \langle s, w\rangle$ imply $\sigma \in \Sigma$ for all s and w?

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, GW91]. 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.

We finally present a class of eventuality properties:

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

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

4 Algorithms

In this section we give algorithms for deciding the reachability problem and the eventuality problem. Safety properties can be verified from the algorithm for reachability, as described in Section 3.

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 $w_1 \preceq w_2$.

4.1 Deciding the Reachability Problem

The main idea of our algorithm for deciding whether some global state in a set N is reachable is to perform a reachability analysis "backwards" from the set N, 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 $\leadsto$ 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 $\leadsto$. In some cases $\langle s_1, w_1 \rangle \leadsto \langle s_2, w_2 \rangle$ denotes that the state $\langle s_1, w_1 \rangle$ can be reached from $\langle s_2, w_2 \rangle$ by first performing a $\longrightarrow$-transition and thereafter lose a certain message. In any case, we prove in Theorem 4.3 that $\langle s_1, \varepsilon \rangle \xrightarrow{*} \langle s_2, w_2 \rangle$ if and only if $\langle s_2, w_2 \rangle \overset{*}{\leadsto} \langle s_1, \varepsilon \rangle$, showing that we can decide the reachability problem by a backwards reachability analysis from the set N. This backward search is not *a priori* bounded. In order to bound the search, we prove in Lemma 4.4 that if $\langle s_1, w_1 \rangle \preceq \langle s_2, w_2 \rangle$, then for each $\leadsto$-path from $\langle s_2, w_2 \rangle$ to the initial global state there is a shorter or equal-length $\leadsto$-path from $\langle s_1, w_1 \rangle$ 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 4.5) that this fact bounds the number of states that must be analyzed.

Definition 4.1 Let $\mathcal{L} = \langle S, s_0, A, C, M, \delta \rangle$ be a lossy channel system. Define $\leadsto$ 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 $\langle s_1, w\rangle \rightsquigarrow \langle s_2, w\rangle$. $\square$

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

Lemma 4.2 *If $\langle s_1, w_1\rangle$ and $\langle s_2, w_2\rangle$ are global states of a lossy channel system, then*

(a) $(\langle s_1, w_1\rangle \rightsquigarrow \langle s_2, w_2\rangle) \supset \left(\langle s_2, w_2\rangle \xrightarrow{*} \langle s_1, w_1\rangle\right)$

(b) for any string vector w_4 we have

$$(\langle s_1, w_1\rangle \longrightarrow \langle s_2, w_2\rangle \ \wedge \ w_4 \preceq w_2) \supset \left(\exists w_3.[w_3 \preceq w_1 \ \wedge \ \langle s_2, w_4\rangle \overset{*}{\rightsquigarrow} \langle s_1, w_3\rangle]\right)$$

(c) $((\langle s_1, w_1\rangle \rightsquigarrow \langle s_2, w_2\rangle) \ \wedge \ (w_3 \preceq w_1)) \supset \exists w_4.\ (w_4 \preceq w_2) \ \wedge \ (\langle s_1, w_3\rangle \rightsquigarrow \langle s_2, w_4\rangle)$

From Lemma 4.2 we can infer that $\langle s_1, w_1\rangle \overset{*}{\rightsquigarrow} \langle s_2, w_2\rangle$ implies $\langle s_2, w_2\rangle \xrightarrow{*} \langle s_1, w_1\rangle$ (this follows from (a)), and that $\langle s_1, w_1\rangle \xrightarrow{*} \langle s_2, w_2\rangle$ implies that there is a state $\langle s_3, w_3\rangle$ with $\langle s_3, w_3\rangle \preceq \langle s_1, w_1\rangle$ such that $\langle s_2, w_2\rangle \overset{*}{\rightsquigarrow} \langle s_1, w_1\rangle$ (this follows from (b)). In particular, we have the following.

Theorem 4.3 *If $\langle s_1, \varepsilon\rangle$ and $\langle s_2, w_2\rangle$ are global states of a lossy channel system, then*

$$\langle s_1, \varepsilon\rangle \xrightarrow{*} \langle s_2, w_2\rangle \qquad \textit{iff} \qquad \langle s_2, w_2\rangle \overset{*}{\rightsquigarrow} \langle s_1, \varepsilon\rangle$$

Define the *distance* of a global state $\langle s, w\rangle$ (denoted $dist(s, w)$) as the minimal number of $\rightsquigarrow$-transitions needed for coming from $\langle s, w\rangle$ to the initial state $\langle s_0, \varepsilon\rangle$. If the initial state is not reachable from $\langle s, w\rangle$ via $\rightsquigarrow$ then define $dist(s, w) = \infty$. By property *(c)* of Lemma 4.2 we can now infer

Lemma 4.4 *If $\langle s_1, w_1\rangle$ and $\langle s_2, w_2\rangle$ are global states of a lossy channel system such that $\langle s_1, w_1\rangle \preceq \langle s_2, w_2\rangle$ then $dist(s_1, w_1) \leq dist(s_2, w_2)$.*

We are now ready to present the reachability algorithm.

The algorithm inputs the set N of global states, and should check whether N is reachable or not. The algorithm maintains a set W, initialized to N, 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 invariant that W is reachable if and only if N is reachable. Thus, if W becomes empty, then the algorithm terminates concluding that N is unreachable. Otherwise, the algorithm proceeds by analyzing each state in W in turn. When a state $\langle s, w\rangle$ in W is analyzed, three possibilities arise:

1. if $\langle s, w\rangle$ is the initial state then terminate and say that N is reachable,

2. if there exists a state $\langle s', w'\rangle$ in V with $\langle s', w'\rangle \preceq \langle s, w\rangle$, then simply discard $\langle s, w\rangle$ (since Lemma 4.4 implies that if there is a $\leadsto$-path from $\langle s, w\rangle$ to the initial state then there is also a $\leadsto$-path from $\langle s', w'\rangle$ to the initial state which does not pass $\langle s, w\rangle$),

3. otherwise generate the $\leadsto$-successors of $\langle s, w\rangle$, put these into W, and move $\langle s, w\rangle$ from W to V. Furthermore, remove each member $\langle s', w'\rangle$ of V for which $\langle s, w\rangle \preceq \langle s', w'\rangle$ (By Lemma 4.4 this does not affect the information contained in V).

The reason why the algorithm always terminates is that the set V cannot grow without bound. This can be explained as follows. Whenever a new element $\langle s, w\rangle$ is added to V it is ensured that $\langle s', w'\rangle \not\preceq \langle s, w\rangle$, for each $\langle s', w'\rangle$ already added to V. This means that the sequence of global states added to V forms a sequence

$$\langle s_1, w_1\rangle \ \langle s_2, w_2\rangle \ \langle s_3, w_3\rangle \ \cdots$$

such that $\langle s_i, w_i\rangle \not\preceq \langle s_j, w_j\rangle$ 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 4.5 *(Higman's theorem) There is no infinite sequence of global states*

$$\langle s_1, w_1\rangle \ \langle s_2, w_2\rangle \ \langle s_3, w_3\rangle \ \cdots$$

such that $\langle s_i, w_i\rangle \not\preceq \langle s_j, w_j\rangle$ *for all* $i < j$.

4.2 Deciding the Eventuality Problem

An algorithm for deciding whether each execution eventually reaches a state in a certain set N can be obtained by performing a simple reachability analysis (using the relation $\longrightarrow$) from the initial state. Starting from the initial state $\langle s_0, \varepsilon\rangle$, a tree of reachable global states is constructed. A branch of the tree need not be expanded further if it contains a state in N. The algorithm can now end in one of two ways:

1. If all branches of the tree end in a state in N, then the eventuality property holds.

2. If a state $\langle s, w'\rangle$ is a descendant of a state $\langle s, w\rangle$ and N does not contain any state $\langle s, w''\rangle$ with $w \prec w'' \prec w'$, then a loop in the execution has been discovered, which means that the eventuality property does not hold.

Using Theorem 4.5, we can also for this algorithm prove that the size of the reachability tree is bounded and that the algorithm therefore terminates.

5 Complexity

We have not been able to estimate the complexity of the algorithms of Section 4. We suspect that practical examples will have a much lower complexity than the actual worst-case complexity calculated from the size of the examples. As empirical experiments, we have analyzed some sliding-window protocols using a model both unbounded and lossy channels. For these protocols, we have verifed the safety properties that they synchronize with the environment in the same way as a buffer with the appropriate capacity. At the end of the analysis, the set of unreachable states that prove the safety property contains for the Alternating Bit protocol 45 global states, and for a sliding window protocol with 3 different sequence numbers 350 global states.

6 Conclusion

In this paper, we have shown that 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 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 expressions that describes the possible corresponding contents of the channels. We 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 to prove decidability. Pachl gives no algorithm for constructing a description of the reachable states in the case 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 [Cou91].

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.

[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. *J. ACM*, 2(5):323–342, April 1983.

[CF87] A. Choquet and A. Finkel. Simulation of linear FIFO nets having a structured set of terminal markings. In *Proc. 8th European Workshop on Applications and Theory of Petri Nets*, 1987.

[Cou91] B. Courcelle. On constructing obstrution 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.

[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.

[GW91] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. In *Proc. Workshop on Computer Aided Verification*, volume 575 of *Lecture Notes in Computer Science*, pages 332–342. Springer Verlag, 1991.

[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.

[JP89] B. Jonsson and J. Parrow. Deciding bisimulation equivalences for a class of non-finite-state programs. In Monien and Cori, editors, *Proc. 6th Symposium on Theoretical Aspects of Computer Science*, volume 349 of *Lecture Notes in Computer Science*, pages 421–433. Springer Verlag, 1989. Revised version to appear in Information and Computation.

[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.

[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, emtpy channel detection and synchronization for communicating finite automata. *Theoretical Computer Science*, 44:69–105, 1986.

[SZ91] A.P. Sistla and L.D. Zuck. Automatic temporal verification of buffer systems. In *Proc. Workshop on Computer Aided Verification*, pages 93–103, 1991.

[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] 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.