Undecidable Verification Problems for Programs with Unreliable Channels

Parosh Aziz 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. In an earlier paper, we showed that several interesting verification problems are decidable for this class of systems, namely (1) the *reachability problem*: is a set of states reachable from some other state of the system, (2) *safety property 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. In this paper, we show that the following problems are undecidable, namely

- The *model checking* problem in propositional temporal logics such as Propositional Linear Time Logic (PTL) and Computation Tree Logic (CTL).
- The problem of deciding *eventuality properties with fair channels*: do all computations eventually reach a given set of states if the unreliable channels are fair in the sense that they deliver infinitely many messages if infinitely many messages are transmitted. This problem can model the question of whether a link protocol, such as HDLC, will eventually reliably transfer messages across a medium that is not permanently broken.

The results are obtained through a reduction from a variant of Post's Correspondence Problem.

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 (e.g. [BCM$^+$90, CES86, VW86]), and successfully applied to e.g. communication protocols (e.g. [Hol91]) and hardware structures. 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, e.g. certain types of real-time systems that operate on clocks [ACD90, Yi91, Cer92], data-independent

*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.

systems [JP89, Wol86], and systems with many identical processes [CG87, GS92, SG90]. In order to extend the applicability of algorithmic verification, we consider it important to investigate whether automatic verification techniques can be developed also for other classes of infinite-state systems.

Recently, the authors 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 [AJ93]. This result was 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. [BZ83]). Lossy channel systems 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 our previous work [AJ93], we defined a model for lossy channel systems, where the global state of the system is described as a pair consisting of a control state (of the finite-state system) and the contents of the channels. We showed that the following verification problems for lossy channel systems are decidable:

- The *reachability problem*: is a set of given states of such a system reachable from some other state of the system.
- *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.
- *Eventuality properties*: Do all computations of a system eventually reach a given set of states. This result has also been proven independently by Finkel [Fin89].

In this paper, we investigate the decidability of more general verification problems for the same class of systems. It might be expected that the techniques used for proving decidability of the previously mentioned verification problems could be extended e.g. to general model-checking of temporal logic formulas. To our surprise, we are now able to prove that most of the interesting verification problems that were not proven decidable in [AJ93] are in fact undecidable. More precisely, we show that the following problems are undecidable:

- The *model checking* problem in propositional temporal logics such as Propositional Linear Time Logic (PTL) and Computation Tree Logic (CTL), interpreted over lossy channel systems. Model checking is more general than the problem of deciding safety properties, which was shown decidable in [AJ93].
- The problem of deciding *eventuality properties with fair channels*: Do all computations eventually reach a given set of states if the unreliable channels are fair in the sense that they deliver infinitely many messages if infinitely many messages are transmitted. This problem is interesting, since one of its instances models the problem of verifying that a link protocol, such as HDLC, will eventually transfer all messages if the unreliable channel is not permanently broken. In [AJ93] we proved that the problem of checking eventuality properties is decidable if no assumptions on fairness in the channels are made. However, without fairness assumptions, the eventuality problem becomes rather uninteresting since most eventuality problems are then trivially false, due to the possibility of always losing all messages in the channels.

To derive the undecidability results we define a problem, called the *recurrent state problem* (RSP), and prove that it is undecidable for lossy channel systems. RSP is the problem of deciding, for a lossy channel system and a control state, whether the system has an infinite computation that visits the control state infinitely often. Visiting a control state infinitely often can be thought of as a fairness property, and thus RSP is a problem which checks a specific fairness property for lossy channel systems.

The fact that RSP corresponds to one particular kind of fairness property makes it possible to reduce RSP to many problems for lossy channel systems that involve fairness. Examples of such problems are model checking of formulas in a propositional temporal logic that can express fairness and liveness, or the problem of checking whether a certain eventuality problem holds if the channels are assumed to be fair.

To explain why RSP can be undecidable, although the problem of checking safety properties is decidable, we offer the following intuitive argument. For lossy channel systems, RSP concerns checking whether there exists a computation in which some property is satisfied infinitely often. We cannot check this on some finite prefix of the computation, but must consider the entire infinite computation. In our proof of undecidability, we construct systems for which all nondeadlocking computations after some point do not lose any messages from the channels. To check whether a computation of these systems satisfies some property infinitely often, we must therefore consider an infinite tail in which the channels behave as perfect channels. Thus, our construction essentially transforms a problem about lossy channels to a problem about unbounded perfect channels. Since most verification problems involving perfect channels are undecidable, it is understandable that also RSP is undecidable for lossy channel systems. By contrast, checking safety properties involves the construction of invariants without the need for considering infinite sequences of transitions.

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 (e.g. channel alphabets of size one [KM69, RY86], bounded channel languages [GGLR87, CF87], and others [Fin88]). These classes are restrictive and do not cover e.g. the Alternating Bit Protocol. 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* (a language is *recognizable* if it is a finite union of cartesian products of regular languages).

Algorithms that are not decision procedures, and which may or may not succeed for a given system, have been developed by Purushothaman and Peng [PP91] and by Brand and Joyner [BZ83]. 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 processes.

Wolper [Wol86] shows that by using an assumption of data-independence, the problem of proving that a data-independent systems 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 lossy channel systems. In Section 3 we briefly review the definitions of problems that are decidable for systems with lossy channels. In Section 4 we prove RSP undecidable. In Sections 5 we reduce RSP to the problem of model checking in propositional temporal logic. In Section 6 we reduce RSP to the problem of checking eventuality properties with fair channels.

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. For $x, y \in M^*$ we let $x \bullet y$ denote the concatenation of x and y. 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 $\gamma \stackrel{\sigma}{\Longrightarrow} \gamma'$ 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.

A *computation* is an infinite sequence $\gamma_1\ \gamma_2\ \gamma_3\ \cdots$ of global states, where γ_1 is equal to the initial global state γ_0, such that for each i one of the following conditions is fulfilled:

1. $\gamma_i \longrightarrow \gamma_{i+1}$; or
2. $\gamma_i = \gamma_{i+1}$ and there is no γ such that $\gamma_i \longrightarrow \gamma$.

Intuitively the second case means that if we reach a global state γ where the system is "deadlocked", then the computation is completed by repeating γ infinitely many times.

3 Decidable Properties of Lossy Channel Systems

In this section, we present the reachability problem, safety properties, and eventuality properties. These problems are shown to be decidable in [AJ93].

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 any 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. The similar problem of checking whether a finite set T of control states is reachable (i.e. checking whether a global state, whose control component is in T, is reachable) is also decidable.

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 s is a "bad" control state, then the property "s 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. 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 regular 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.

Eventuality Properties A simple class of eventuality properties can be defined as:

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

Question: Do all computations of $\mathcal{L}$ eventually reach a control state in T (i.e. reach a global state whose control component is in T)?

Note that the problem of checking whether all executions of the system eventually terminate can be reduced to the problem of checking the eventuality properties above.

Theorem 3.1 *The reachability problem, safety properties, and eventuality properties are decidable for lossy channel systems.*

Proof: The proof can be found in [AJ93]. □

4 Undecidability of RSP

In this section, we define the problem of checking, for a lossy channel system $\mathcal{L}$ and a control state s in $\mathcal{L}$, whether $\mathcal{L}$ has any computation which visits s infinitely often. We call the problem *the recurrent state problem (RSP)* for lossy channel systems.

We prove (Corollary 4.3) that RSP is undecidable. The undecidability result is achieved through a reduction (described in Theorem 4.2) from the *Cyclic Post's Correspondence Problem*. The Cyclic Post's Correspondence Problem was shown to be undecidable by Ruohonen in [Ruo83] (Theorem 4.1).

The reason for introducing RSP is that it can be reduced to the verification problems of model checking for PTL (Theorem 5.1), model checking for CTL (Theorem 5.2), and checking eventuality properties with fair channels (Theorem 6.1). Hence the undecidability results for these problems follow immediately from the undecidability of RSP.

For a lossy channel system $\mathcal{L}$, a computation π of $\mathcal{L}$, and a control state s of $\mathcal{L}$, we say that π *visits s infinitely often* if there are infinitely many occurrences of global states γ in π such that the control state of γ is s.

The Recurrent State Problem (RSP) The definition of RSP is as follows.

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

Question: Is there a computation of $\mathcal{L}$ visiting s infinitely often?

For $x, y \in M^*$, let $x =_c y$ denote that there are $x_1, x_2 \in M^*$ such that $x = x_1 \bullet x_2$ and $y = x_2 \bullet x_1$. Also, let $x \preceq_c y$ denote that there are $x_1, x_2 \in M^*$ such that $x = x_1 \bullet x_2$ and $x_2 \bullet x_1 \preceq y$. Intuitively $x =_c y$ means that if x and y are considered as "circular stings", i.e. if the first and the last elements of each string are "connected" then they become equal. The relation $x \preceq_c y$ can be explained in a similar manner.

The Cyclic Post's Correspondence Problem (CPCP) The definition of CPCP is the following.

Instance: A finite alphabet M and two ordered lists $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$ of elements in M^*.

Question: Does there exist a finite sequence $i_1, i_2, \ldots, i_m$, with $m \geq 1$ and $1 \leq i_j \leq n$, such that

$$x_{i_1} \bullet x_{i_2} \bullet \cdots \bullet x_{i_m} =_c y_{i_1} \bullet y_{i_2} \bullet \cdots \bullet y_{i_m}$$

Theorem 4.1 *CPCP is undecidable.*

Proof: The proof can be found in [Ruo83]. □

Now we prove the main result of this section.

Theorem 4.2 *CPCP is reducible to RSP.*

Proof Idea: The details of the proof can be found in [AJ94]. Given an instance Π of CPCP where the alphabet is M and the two ordered lists are $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$, we construct a lossy channel system $\mathcal{L}$ with a control state s_1 of $\mathcal{L}$ such that Π has a solution if and only if $\mathcal{L}$ has a computation which visits s_1 infinitely often. The system $\mathcal{L}$ has two channels, which we call c and d. Intuitively, $\mathcal{L}$ simulates Π as follows. In the initial control state s_0 of $\mathcal{L}$, any sequence of messages over M can be sent to the channels c and d. This is used by $\mathcal{L}$ to "guess" a solution of Π, by sending two strings of messages to the channels c and d, containing information about the solution of Π. Then, $\mathcal{L}$ moves to the control state s_1 and "checks" whether the guess provided in s_0 is a solution of Π. This is achieved by n sequences of transitions which leave s_1. The i^{th} sequence of transitions changes the contents of channel c from $y_i \bullet z$ to $z \bullet x_i$ for an arbitrary z, and changes the contents of channel d from $x_i \bullet z'$ to $z' \bullet y_i$ for an arbitrary z'. Now, suppose that $i_1, i_2, \ldots, i_m$ is a solution to Π, let $x = x_{i_1} \bullet x_{i_2} \bullet \cdots \bullet x_{i_m}$ and let $y = y_{i_1} \bullet y_{i_2} \bullet \cdots \bullet y_{i_m}$. According to the definition of $=_c$, there are x', x'', y', and y'' such that $x = x' \bullet x''$, and $y = y' \bullet y''$, and $y = x'' \bullet x'$, and $x = y'' \bullet y'$. A computation of $\mathcal{L}$ visiting s_1 infinitely often is now constructed as follows. First, in the control state s_0, $\mathcal{L}$ sends the string $y \bullet x''$ channel c and the string $x \bullet y''$ to channel d. Then, $\mathcal{L}$ moves to the control state s_1, without changing the contents of the channels. From s_1, the contents $y \bullet x'' = y_{i_1} \bullet \cdots \bullet y_{i_n} \bullet x''$ of channel c and the contents $x \bullet y'' = x_{i_1} \bullet \cdots \bullet x_{i_n} \bullet y''$ of channel d are first transformed into $y_{i_2} \bullet \cdots \bullet y_{i_n} \bullet x'' \bullet x_{i_1}$ and $x_{i_2} \bullet \cdots \bullet x_{i_n} \bullet y'' \bullet y_{i_1}$ respectively, by the i_1^{th} sequence; thereafter the channel contents are transformed again by the i_2^{th} sequence , etc. until finally c contains $x'' \bullet x_{i_1} \bullet \cdots \bullet x_{i_n} = x'' \bullet x$ and d contains $y'' \bullet y_{i_1} \bullet \cdots \bullet y_{i_n} = y'' \bullet y$. Since $x'' \bullet x = y \bullet x''$ and $y'' \bullet y = x \bullet y''$, we are back at the beginning (at control state s_1) and can repeat the checking cycle an infinite number of times, thus visiting s_1 infinitely many times. For the proof in the other direction, it can be shown that if Π has no solutions, then each computation reaching s_1 will eventually deadlock without visiting s_1 infinitely many times. $\square$

Corollary 4.3 *RSP is undecidable.*

Proof: The result follows directly from Theorem 4.1 and Theorem 4.2. $\square$

Since only two channels are used in the definition of the lossy channel system in Theorem 4.2, the proof of the theorem shows that RSP is undecidable for the class of lossy channel systems with at least two channels. The undecidability result holds even for the class of lossy channel systems with only one channel. We define n-RSP to be RSP when restricted to the class of systems with at most n channels. We know from Corollary 4.3 that 2-RSP is undecidable.

Theorem 4.4 *1-RSP is undecidable.*

Proof Idea: The details of the proof can be found in [AJ94]. The main idea is that, given a lossy channel system $\mathcal{L}$ (with two channels), we construct a lossy channel system $\mathcal{L}'$ (with one channel) which "simulates" $\mathcal{L}$. Each global state γ of $\mathcal{L}$ is "simulated" by a global state γ' of $\mathcal{L}'$, where the control states of γ and γ' are the same, and where the channel content in γ' is a merge of the contents of the two channels in γ. To simulate the reception of messages which are not at the head of the channel of $\mathcal{L}'$, we perform a "rotation operation". This means that the content of the channel is "rotated" until the message is obtained at the head of the channel, and then the rotation is resumed until the original ordering of messages inside the channel is obtained again. $\square$

5 Propositional Temporal Logic

In this section we define *Propositional Linear Time Logic (PTL)*, interpreted over lossy channel systems, and then prove that it is undecidable to check whether a lossy channel system satisfies a PTL formula. We indicate also how similar undecidability results may be obtained for *Computation Tree Logic (CTL)*.

Formulas of PTL are built from a set *Prop* of *atomic propositions*, and are closed under the application of Boolean connectives, the unary temporal connective $\bigcirc$ (next), and the binary temporal connective $\mathcal{U}$ (until). For a computation π, let $\pi(i)$ denote the i^{th} element of π. Given a lossy channel system $\mathcal{L}$, with a set of control states S, a *labeling function* is a mapping $f : S \to 2^{Prop}$, which assigns truth values to the elements of *Prop* at each control state of S. For a lossy channel system $\mathcal{L}$, a labeling function f, a computation π of $\mathcal{L}$, a point $i \geq 0$, and PTL formulas ϕ and ψ, we have that:

$$
\begin{array}{rcl}
f, \pi, i \models p, \text{ for } p \in Prop & \text{iff} & p \in f(s), \text{ where } s \text{ is the control state of } \pi(i) \\
f, \pi, i \models \phi \wedge \psi & \text{iff} & f, \pi, i \models \phi \text{ and } f, \pi, i \models \psi. \\
f, \pi, i \models \neg\phi & \text{iff} & f, \pi, i \not\models \phi. \\
f, \pi, i \models \bigcirc\phi & \text{iff} & f, \pi, i+1 \models \phi. \\
f, \pi, i \models \phi \,\mathcal{U}\, \psi & \text{iff} & \text{for some } j \geq i, \text{ we have } f, \pi, j \models \psi, \\
 & & \text{and for all } k \text{ with } i \leq k < j, \text{ we have } f, \pi, k \models \phi.
\end{array}
$$

We define $\Diamond\phi = true\ \mathcal{U}\ \phi$, and $\Box\phi = \neg\Diamond\neg\phi$. We say that π *satisfies* ϕ *under* f, denoted $f, \pi \models \phi$, iff $f, \pi, 0 \models \phi$. We say that $\mathcal{L}$ *satisfies* ϕ *under* f, denoted $f, \mathcal{L} \models \phi$, iff for each computation π of $\mathcal{L}$ we have $f, \pi \models \phi$.

Model Checking for PTL The model checking problem for PTL, interpreted over lossy channel systems, is defined as follows.

Instance: A lossy channel system $\mathcal{L}$, a labeling function f, and a PTL formula ϕ.

Question: $f, \mathcal{L} \models \phi$?

Theorem 5.1 *The model checking problem for PTL interpreted over lossy channel systems is undecidable.*

Proof: The proof is achieved by reducing RSP to the model checking problem. Given a lossy channel system $\mathcal{L}$ and a control state s in $\mathcal{L}$, we define a labeling function f and a PTL formula ϕ such that $f, \mathcal{L} \not\models \phi$ if and only if $\mathcal{L}$ has a computation which visits s infinitely many times. We define f such that there is a proposition p, where $p \in f(s)$ and $p \notin f(s')$ for any $s' \neq s$. The formula ϕ is defined as $\Diamond\Box\neg p$. $\Box$

Computation tree Logic (CTL) can be defined for lossy channel systems in a similar manner to PTL, and the following result can be obtained.

Theorem 5.2 *The model checking problem for CTL interpreted over lossy channel systems is undecidable.*

Proof: The proof is similar to that of Theorem 5.1. Here, the corresponding CTL formula is defined as $AF(AG(\neg p))$. $\Box$

6 Eventuality Properties with Fair Channels

In this section, we consider the verification of eventuality properties of the form: do all computations eventually reach a given set of states. Eventuality properties are often interesting only under the assumption that the channels are not permanently broken. Such an assumption is usually formalized as a fairness property on the channels. We first define a fairness condition on message losses in channels, namely if a message is infinitely often sent to a channel, then it is infinitely often received from the channel. We study a variant of the eventuality properties (described in Section 3), where the problem is formulated over the set of *fair computations*, i.e. computations which fulfill the fairness property, instead of the set of all computations of the system. We prove that eventuality properties become undecidable under our fairness condition. To show that the undecidability result is robust under variations of the formulation of the fairness property, we introduce (Section 6.2) alternative fairness conditions. It is easy to show that simple modifications of the undecidability proof for the first fairness condition give undecidability proofs for the eventuality properties under the alternative fairness conditions.

6.1 A Fairness Condition

We define the first fairness condition called *impartiality* and prove (Theorem 6.1) that the eventuality properties are undecidable under the impartiality condition. For a lossy channel system $\mathcal{L} = \langle S, s_0, A, C, M, \delta \rangle$, a computation $\pi = \gamma_1\ \gamma_2\ \gamma_3\ \cdots$, where γ_i is of the form $\langle s_i, w_i \rangle$, is said to be *impartial with respect to a channel* $c \in C$ *and a message* $m \in M$ if and only if the following holds: if there are infinitely many steps in π where m is sent to c, then there are infinitely many steps where m is received from c. Equivalently, if there are infinitely many i such that $w_{i+1} = w_i[c := w_i(c) \bullet m]$, then there are infinitely many i such that $\langle s_i, c?m, s_{i+1} \rangle \in \delta$ and $w_i = w_{i+1}[c := m \bullet w_{i+1}(c)]$. The computation π is said to be *impartial with respect to a channel* $c \in C$ if and only if it is impartial with respect to c and all messages in M. The computation π is said to be *impartial* if it is impartial with respect to all channels in C.

Eventuality Properties under Impartiality We re-formulate the the eventuality properties (described in Section 3), and consider the following problem.

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

Question: Do all impartial computations of $\mathcal{L}$ eventually reach a state in T?

Theorem 6.1 *The eventuality properties under impartiality are undecidable for lossy channel systems.*

Proof Idea: The details of the proof can be found in [AJ94]. The proof is achieved by reducing RSP to the eventuality properties under impartiality. Given a lossy channel system $\mathcal{L}$ and a control state r of $\mathcal{L}$, we construct a lossy channel system $\mathcal{L}'$, and a control state t of $\mathcal{L}'$ (where t does not belong to the control states of $\mathcal{L}$), such that $\mathcal{L}$ has a computation which visits r infinitely often if and only if $\mathcal{L}'$ has an impartial computation which does not reach t. Without loss of generality (Theorem 4.4) we assume that $\mathcal{L}$ has only one channel. The main idea is that we add a new message ℓ (not belonging to the set of messages of $\mathcal{L}$) to the set of messages of $\mathcal{L}'$. A computation π of $\mathcal{L}$ which visits r infinitely often is "simulated" by an impartial computation π' of $\mathcal{L}'$ which does not reach t. We construct $\mathcal{L}'$ such that π' has the following properties: (1) each message loss in π is replaced by a message reception in π' (this means that π' is impartial with respect to the messages of $\mathcal{L}$, and we only need to worry about the impartiality of π' with respect to the message ℓ), (2) ℓ is sent to the channel infinitely often, and (3) the only control state where ℓ is received is r. Furthermore, each computation of $\mathcal{L}'$ deadlocks if and only if it reaches t.

Now if $\mathcal{L}$ has a computation π visiting r infinitely many times, then it can be simulated by a computation π' of $\mathcal{L}'$ also visiting r infinitely many times. Each time π' reaches r it receives the message ℓ (the construction of π' is such that ℓ is always at the head of the channel when π' reaches r). It is clear that π' is impartial and that it will not reach t. On the other hand, if $\mathcal{L}'$ has an impartial computation π' not reaching t, then by the construction of $\mathcal{L}'$ it will never deadlock, and it sends the message ℓ infinitely often to the channel. It follows that π' (and consequently π) visits r infinitely often, since π' is impartial and r is the only control state where ℓ can be received. $\square$

6.2 Other Fairness Conditions

We define two new fairness conditions on message losses. It is easy to show that simple modifications of the proof of Theorem 6.1 give undecidability proofs for the eventuality properties under the new fairness conditions.

1. For a lossy channel system $\mathcal{L} = \langle S, s_0, A, C, M, \delta \rangle$, a computation $\pi = \gamma_1\ \gamma_2\ \gamma_3\ \cdots$, where γ_i is of the form $\langle s_i, w_i \rangle$, is said to be *message fair with respect to a channel $c \in C$ and a message $m \in M$* if and only if the following holds: if there are infinitely many steps in π where m is sent to c, and if there are infinitely many i where $\langle s_i, c?m, s_i' \rangle \in \delta$, for some $s_i' \in S$, then there are infinitely many steps where m is received from c.
2. For a lossy channel system $\mathcal{L} = \langle S, s_0, A, C, M, \delta \rangle$, a computation $\pi = \gamma_1\ \gamma_2\ \gamma_3\ \cdots$ is said to be *channel fair with respect to a channel $c \in C$* if and only if the following holds: if there are infinitely many steps in π where a message is sent to c, then there are infinitely many steps where a message is received from c.

Conclusions

We have shown the undecidability of several types of verification problems that involve fairness or liveness properties of systems of finite-state processes that communicate over unbounded but lossy FIFO channels. Our results indicate that most problems that involve proving some kind of fairness property are undecidable. This is in contrast to our earlier results that showed safety properties to be decidable for the same class of system.

A way of making these results concrete is to consider a protocol such as HDLC, which is designed to transmit messages correctly over lossy unbounded FIFO channels. Using general verification algorithms for lossy channel systems, we can automatically verify the absence of incorrect message deliveries, but we cannot automatically prove that all messages are eventually delivered correctly if the channel is not permanently broken.

Acknowledgements

We are grateful to Alain Finkel, S. Purushothaman, and Yih-Kuen Tsay for comments and discussions. We thank Karlis Čerāns for bringing to our attention the undecidability proof of the Cyclic Post's Correspondence Problem in [Ruo83].

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.

[AJ93] Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. In *Proc. 8^{th} IEEE Int. Symp. on Logic in Computer Science*, 1993.

[AJ94] Parosh Aziz Abdulla and Bengt Jonsson. Undecidable verification problems for programs with unreliable channels. Technical Report 94/47, Dept. of Computer Systems, Uppsala University, Sweden, 1994.

[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.

[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.

[Cer92] K. Cerans. 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.

[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.

[Fin89] A. Finkel. Decidability of the termination problem for completely specified protocols. Technical Report 463, LRI, University Paris-Sud, 1989. Also accepted for publication in the Distributed Computing Journal.

[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.

[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.

[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.

[Ruo83] K. Ruohonen. On some variants of Post's correspondence problem. *Acta Informatica*, (19):357–367, 1983.

[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.

[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.