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 the problems of checking reachability, safety properties, and eventuality properties are decidable for this class of systems. 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 Temporal 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 satisfy fairness assumptions.

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

*This research report is a revised and extended version of a paper that has appeared under the same title in the proceedings of the 21st International Colloquium on Automata, Languages and programming (ICALP), 1994, published as vol. 820 of Lecture Notes in Computer Science, Springer Verlag, pp. 316-327.

†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 and 95-796.

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 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, Č92, LY93], data-independent systems [JP93, Wol86], systems with many identical processes [CG87, GS92, SG90], context-free processes ([BS92, CHS92, CHM93]), and Petri nets ([Jan90]).

Recently, the authors of this paper 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]. The HDLC protocol is intended to provide reliable full-duplex data transfer between protocol entities, using an error-prone physical communication channel. The protocol includes procedures both for communication management and data transfer. These protocols and others are designed to operate correctly even in the case that the FIFO channels are faulty and may lose messages. 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*: do all computations of a system stay within a *regular set* of allowed finite traces. This problem can be verified via a transformation to the reachability problem.
- *Eventuality properties*: Do all computations of a system eventually reach a given set of states. This result has also been proven independently by Finkel [Fin94].

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 satisfy some fairness assumptions. There are several kinds of fairness assumptions, which we consider in this paper. A typical one is that each channel delivers infinitely many messages if infinitely many messages are transmitted to it. This problem has significant practical interest, 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 channels 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, most eventuality problems are often 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 undecidability of RSP is shown through a reduction from a variant of the Post's Correspondence Problem called the *Cyclic Post's Correspondence Problem* [Ruo83]. As far as we are aware, this is the first application of the Cyclic Post's Correspondence Problem to prove the undecidabilty of a property for infinite traces.

The fact that RSP corresponds to one particular kind of fairness property makes it possible to reduce RSP to many problems for lossy channel sys-

tems 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 in which there are computation satisfying a certain property infinitely often if and only there are perfect computations (i.e. computations which do not lose any messages) which satisfy the property infinitely often. 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 $x \in M^*$, let $x(i)$ denote the i^{th} element of x. 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 $\gamma \stackrel{a}{\longrightarrow} \gamma'$ 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 $\langle s_1, w \rangle \stackrel{\tau}{\longrightarrow} \langle s_2, w[c := w(c) \bullet m] \rangle$, 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 $\langle s_1, w[c := m \bullet w(c)] \rangle \stackrel{\tau}{\longrightarrow} \langle s_2, w \rangle$, 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 the computation reaches 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 the Reccurent State Problem

In this section, we describe 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 4.7) that RSP is undecidable. The undecidability result is achieved through a reduction (described in Theorem 4.6) from the *Cyclic Post's Correspondence Problem*, which was shown to be undecidable by Ruohonen in [Ruo83] (Theorem 4.5).

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 from the undecidability of RSP.

4.1 Preliminaries

To prove the main result of the section, we need the following definitions and lemmas on strings.

For $x, y \in M^*$, let $x \preceq y$ denote that x is a (not necessarily contiguous) substring of y. Let $x =_c y$ denote that there are $x_1, x_2 \in M^*$ such that $x = x_1 \bullet x_2$ and $y = x_2 \bullet x_1$; and $x \preceq_c y$ 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.

Lemma 4.1 *For any finite set M, and $x, y, z \in M^*$, we have*

$$(x \preceq_c y) \ \wedge \ (y \preceq_c x) \ \supset \ (x =_c y)$$

Proof: The result follows immediately from the fact that $x \preceq_c y$ and $y \preceq_c x$ imply that $|x| = |y|$. $\Box$

Lemma 4.2 *For any finite set M, and $x, y, z \in M^*$, we have*

$$(x \bullet z \preceq z \bullet y) \ \supset \ (x \preceq_c y)$$

Proof: By induction on $|z|$.

Base Case: If $|z| = 0$ then $z = \varepsilon$, and the proof is trivial.

Induction Step: If $x = \varepsilon$ then the proof is trivial. If $x \neq \varepsilon$, there are two cases.

1. There are x_1 and x_2 such that $x = x_1 \bullet x_2$, $x_1 \preceq z$, and $x_2 \bullet z \preceq y$. This means that $x_2 \bullet x_1 \preceq y$. From the definition of $\preceq_c$ it follows that $x \preceq_c y$.

2. There are z_1 and z_2 such that $z = z_1 \bullet z_2$, with $x \bullet z_1 \preceq z$ and $z_2 \preceq y$. It follows that $x \bullet z_1 \preceq z_1 \bullet z_2$ whence $x \bullet z_1 \preceq z_1 \bullet y$. We observe that $z_2 \neq \varepsilon$, otherwise $x \bullet z_1 \preceq z = z_1$, which is a contradiction since $x \neq \varepsilon$. Consequently $|z_1| < |z|$. From the induction hypothesis it follows that $x \preceq_c y$. $\square$

Corollary 4.3 *For any finite set M, and $x, y, z_1, z_2 \in M^*$, we have*

$$[(z_1 \preceq z_2) \ \wedge \ (x \bullet z_2 \preceq z_1 \bullet y)] \ \supset \ (x \preceq_c y)$$

Proof: Suppose that $z_1 \preceq z_2$ and $x \bullet z_2 \preceq z_1 \bullet y$. It follows that $x \bullet z_2 \preceq z_2 \bullet y$. From Lemma 4.2 we get $x \preceq_c y$. $\square$

Theorem 4.4 *(Higman's theorem) Let M be a finite set. There is no infinite sequence $w_1\ w_2\ w_3\ \cdots$ of strings in M^*, such that $w_i \not\preceq w_j$ for all $i < j$.*

The proof of the theorem can be found e.g. in [Hig52] and [Cou91]. It is straightforward to generalize the theorem to sequences $\gamma_1\ \gamma_2\ \gamma_3\ \cdots$ of global states.

4.2 The Recurrent State Problem (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 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?

4.3 The Cyclic Post's Correspondence Problem

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 4.5 *CPCP is undecidable.*

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

4.4 Undecidability of RSP

We prove the main result of this section.

Theorem 4.6 *CPCP is reducible to RSP.*

Proof : First, we give an overview of the proof. 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 is. This can be 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 all of which start and end in s_1. The i^{th} sequence of transitions changes the contents of channel c by removing the string y_i from the front and appending the string x_i to the end, and it changes the contents of channel d by removing the string x_i from the front and appending the string y_i to the end. 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 $y = x'' \bullet x'$. 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 $y \bullet x''$ into channel c and the string $x \bullet x'$ to channel d. Then, $\mathcal{L}$ moves to the control state s_1, without changing the contents of the channels. From s_1, the system $\mathcal{L}$ performs first the i_1^{th} sequence, then the i_2^{th} sequence of transitions, etc. until finally the i_n^{th} sequence of transitions has been performed. The result of these sequences of transitions is that the contents of c are transformed from $(x'' \bullet x') \bullet x'' = (y_{i_1} \bullet \cdots \bullet y_{i_n}) \bullet x''$ into $x'' \bullet (x_{i_1} \bullet \cdots \bullet x_{i_n}) = x'' \bullet (x' \bullet x'')$, and that the contents of d are transformed from $(x' \bullet x'') \bullet x' = (x_{i_1} \bullet \cdots \bullet x_{i_n}) \bullet x'$ into $x' \bullet (y_{i_1} \bullet \cdots \bullet y_{i_n}) = x' \bullet (x'' \bullet x')$. We are thus back at the beginning (at control state s_1 and channel contents $y \bullet x''$ and $x \bullet x'$) and can repeat the sequence of transition sequences 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.

The details of the proof are as follows. Suppose that we are given the instance Π of CPCP as described above. We construct the lossy channel system $\mathcal{L}$ and define the control state s_1 as follows (see[1] Fig. 1). Let $\mathcal{L} = \langle S, s_0, A, C, M, \delta \rangle$, where

- $S = \{s_0, s_1\} \cup S_1 \cup \cdots \cup S_n$, where

  $$\begin{array}{rl} S_i = & \{s_{i,j};\ 1 \leq j \leq |x_i|\} \quad \cup \\ & \{s'_{i,j};\ 1 \leq j \leq |y_i|\} \quad \cup \\ & \{s''_{i,j};\ 1 \leq j \leq |y_i|\} \quad \cup \\ & \{s'''_{i,j};\ 1 \leq j < |x_i|\} \end{array}$$

- $A = \emptyset$.

- $C = \{c, d\}$.

[1] To increase readability, we have not shown all control states and transitions of $\mathcal{L}$ in the figure. The loop from s_0 to s_0, labeled by $\forall m \in M.\ c!m$ and $\forall m \in M.\ d!m$, stands for all transitions in the set δ_1. Also, all "intermediate" transitions in $\delta_2, \ldots, \delta_5$ are dropped. For example, the arrow from s_1 to $s_{1,|x_1|}$, labeled by $c!x_1$, stands for the sequence of transitions $\langle s_1, c!x_1(1), s_{1,1} \rangle, \langle s_{1,1}, c!x_1(2), s_{1,2} \rangle, \ldots, \langle s_{1,|x_1|-1}, c!x_1(|x_1|), s_{1,|x_1|} \rangle$.

- The set δ contains agroup δ_1 and for each i with $1 \leq i \leq n$ four groups $\delta_{i2}, \ldots, \delta_{i5}$ of transitions.

$$\delta_1 : \begin{cases} \langle s_0, c!m, s_0 \rangle, \text{ and } \langle s_0, d!m, s_0 \rangle, \text{ for each } m \in M; \text{ and} \\ \langle s_0, \tau, s_1 \rangle. \end{cases}$$

$$\delta_{i2} : \begin{cases} \langle s_1, c!x_i(1), s_{i,1} \rangle; \\ \langle s_{i,j}, c!x_i(j+1), s_{i,j+1} \rangle, \text{ for each } j \text{ with } 1 \leq j < |x_i|. \end{cases}$$

$$\delta_{i3} : \begin{cases} \langle s_{i,|x_i|}, c?y_i(1), s'_{i,1} \rangle; \\ \langle s'_{i,j}, d?y_i(j+1), s'_{i,j+1} \rangle, \text{ for each } j \text{ with } 1 \leq j < |y_i|. \end{cases}$$

$$\delta_{i4} : \begin{cases} \langle s'_{i,|y_i|}, d!y_i(1), s''_{i,1} \rangle; \\ \langle s''_{i,j}, d!y_i(j+1), s''_{i,j+1} \rangle, \text{ for each } j \text{ with } 1 \leq j < |y_i|. \end{cases}$$

$$\delta_{i5} : \begin{cases} \langle s''_{i,|y_i|}, d?x_i(1), s'''_{i,1} \rangle, \text{ whenever } |x_i| > 1; \\ \langle s'''_{i,j}, d?x_i(j+1), s'''_{i,j+1} \rangle, \text{ for each } j \text{ with } 1 \leq j < |x_i| - 1; \\ \langle s'''_{i,|x_i|-1}, d?x_i(|x_i|), s_1 \rangle, \text{ whenever } |x_i| > 1; \\ \langle s''_{i,|y_i|}, d?x_i(1), s_1 \rangle, \text{ whenever } |x_i| = 1. \end{cases}$$

We explain intuitively how the different groups, δ_1 and $\delta_{i2}, \ldots, \delta_{i5}$ for $1 \leq i \leq n$ allow us to perform certain sequences of transitions.

1. The transitions in δ_1 mean that from the initial control state s_0 we can send any strings of messages to channel c and channel d. Then we can move to the control state s_1 without changing the channels.

2. The transitions in δ_{i2} mean that we can move from s_1 to $s_{i,|x_i|}$, while appending the string x_i to channel c.

3. The transitions in δ_{i3} mean that, if y_i is a substring of the content of channel c, then we can move from $s_{i,|x_i|}$ to $s'_{i,|y_i|}$, while removing the string y_i from channel c (losing messages in c if necessary).

4. The transitions in δ_{i4} mean that we can move from $s'_{i,|y_i|}$ to $s''_{i,|y_i|}$, while appending the string y_i to channel d.

5. The transitions in δ_{i5} mean that, if x_i is a substring of the content of channel d, then we can move from $s''_{i,|y_i|}$ to s_1, while removing the string x_i from channel d (losing messages in d if necessary).

We represent global states of $\mathcal{L}$ as triples $\langle s, x, y \rangle$, where s is the control state and $x, y \in M^*$ are the contents of the channels c and d respectively.

From the discussion above, we observe that $\langle s_0, \varepsilon, \varepsilon\rangle \xrightarrow{*} \langle s_1, x, y\rangle$, for any $x, y \in M^*$, using the transitions in δ_1, and that $\langle s_1, y_i \bullet z_1, x_i \bullet z_2\rangle \xrightarrow{*} \langle s_1, z_1 \bullet x_i, z_2 \bullet y_i\rangle$, for any $z_1, z_2 \in M^*$, using the transitions in $\delta_{i2}, \ldots, \delta_{i5}$. It follows that for any finite sequence of natural numbers $i_1, i_2, \ldots, i_m$, with $m \geq 1$ and $1 \leq i_j \leq n$, if $x = x_{i_1} \bullet x_{i_2} \bullet \cdots \bullet x_{i_m}$, and if $y = y_{i_1} \bullet y_{i_2} \bullet \cdots \bullet y_{i_m}$, then $\langle s_1, y \bullet z_1, x \bullet z_2\rangle \xrightarrow{*} \langle s_1, z_1 \bullet x, z_2 \bullet y\rangle$, for each $z_1, z_2 \in M^*$.

Now we show that Π has a solution if and only if $\mathcal{L}$ has a computation π which visits s_1 infinitely often.

[only if] Suppose that Π has a solution $i_1, i_2, \ldots, i_m$. It follows that $x_{i_1} \bullet x_{i_2} \bullet \cdots \bullet x_{i_m} =_c y_{i_1} \bullet y_{i_2} \bullet \cdots \bullet y_{i_m}$. Let $x = x_{i_1} \bullet x_{i_2} \bullet \cdots x_{i_m}$ and $y = y_{i_1} \bullet y_{i_2} \bullet \cdots y_{i_m}$. According to the definition of $=_c$, there are x', x'' such that $x = x' \bullet x''$ and $y = x'' \bullet x'$. Also from the construction of $\mathcal{L}$ it follows that $\langle s_1, y \bullet z_1, x \bullet z_2\rangle \xrightarrow{*} \langle s_1, z_1 \bullet x, z_2 \bullet y\rangle$, for each $z_1, z_2 \in M^*$.

The computation π is built as follows.

1. From $\langle s_0,\ \varepsilon,\ \varepsilon\rangle$ we move to $\langle s_1,\ y \bullet x'',\ x \bullet x'\rangle$, using the transitions in δ_1.
2. From $\langle s_1,\ y \bullet x'',\ x \bullet x'\rangle$ we move to $\langle s_1,\ x'' \bullet x,\ x' \bullet y\rangle$, using the transitions in $\delta_{i2}, \ldots, \delta_{i5}$.
3. We know that $x'' \bullet x = x'' \bullet x' \bullet x'' = y \bullet x''$, and that $x' \bullet y = x' \bullet x'' \bullet x' = x \bullet x'$. Therefore, the sequence of transitions in (2) can be repeated infinitely often. This means that π visits s_1 infinitely often.

[if] Suppose that there is a computation π which visits s_1 infinitely often. By Higman's theorem (Theorem 4.4) it follows that there are global states $\gamma_1 = \langle s_1, u_1, v_1\rangle$ and $\gamma_2 = \langle s_1, u_2, v_2\rangle$, such that $\gamma_1 \preceq \gamma_2$ (i.e. $u_1 \preceq u_2$ and $v_1 \preceq v_2$) and $\gamma_1 \xrightarrow{*} \gamma_2$. When moving from γ_1 to γ_2, let the string added to c (d) be z_1 (z_2), and the string received from c (d) be z_3 (z_4). Observe that π cannot deadlock in s_1, and hence $z_1, z_2 \neq \varepsilon$. It is clear that $z_3 \bullet u_2 \preceq u_1 \bullet z_1$, and that $z_4 \bullet v_2 \preceq v_1 \bullet z_2$. From Corollary 4.3 and the fact that $u_1 \preceq u_2$ and $v_1 \preceq v_2$, it follows that $z_3 \preceq_c z_1$ and $z_4 \preceq_c z_2$. Furthermore, from the construction of $\mathcal{L}$ we observe that there are $i_1, \ldots, i_m$ such that $z_1 = z_4 = x_{i_1} \bullet \cdots \bullet x_{i_m}$ and $z_2 = z_3 = y_{i_1} \bullet \cdots \bullet y_{i_m}$. From Lemma 4.1 it follows that $x_{i_1} \bullet \cdots \bullet x_{i_m} =_c y_{i_1} \bullet \cdots \bullet y_{i_m}$. Since $z_1, z_2 \neq \varepsilon$, it follows that $m \geq 1$. $\square$

In fact, we can show an interesting property of the lossy channel system $\mathcal{L}$, namely that Π has a solution (or equivalently $\mathcal{L}$ has a computation visiting s infinitely often) if and only if $\mathcal{L}$ has a perfect computation (i.e. a computation which does not lose any messages) visiting s infinitely often. To prove the claim, we see that the if-direction is obvious, and for the only if-direction, we

note that the computation constructed in the proof of Theorem 4.6 is indeed perfect.

Corollary 4.7 *RSP is undecidable.*

Proof: The result follows directly from Theorem 4.5 and Theorem 4.6. □

4.5 One-Channel Systems

Since only two channels are used in the definition of the lossy channel system in Theorem 4.6, the proof of the theorem shows that RSP is undecidable for the class of lossy channel systems with at least two channels. We show that 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 4.7 that 2-RSP is undecidable.

Theorem 4.8 *1-RSP is undecidable.*

Proof: Given a lossy channel system $\mathcal{L} = \langle S, s_0, A, \{c, d\}, M, \delta \rangle$ (with two channels c and d), with a control state $s \in S$, we construct a lossy channel system $\mathcal{L}' = \langle S \cup S', s_0, A, \{c'\}, M', \delta' \rangle$ (with one channel c') such that $\mathcal{L}$ has a computation which visits s infinitely often if and only if $\mathcal{L}'$ has a computation which visits s infinitely often. The idea of the proof is that we use the channel c' of $\mathcal{L}'$ to "simulate" both channels of $\mathcal{L}$. Without loss of generality we assume that the channels of $\mathcal{L}$ have different alphabets, i.e. the sets of messages sent to the two channels are disjoint. 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. If $\mathcal{L}$ sends a new message to one of its channels, then this operation is simulated by sending the same message to the channel of $\mathcal{L}'$. To simulate receiving a message ℓ from (say) channel c, $\mathcal{L}'$ checks the first message m on c'. If m belongs to the alphabet of c and $m = \ell$, then m is simply received by $\mathcal{L}'$. If m does not belong to the alphabet of c, then $\mathcal{L}'$ moves to a special control state, appends a special symbol $\sharp$ to the end of c', and then starts "rotating" the content of c' (the rotation is achieved by receiving messages from the head of c' and appending them to the tail of c'). The rotation continues until a message m' belonging to the alphabet of c is obtained at head of c'. If $m' \neq \ell$ then $\mathcal{L}'$ deadlocks. If $m' = \ell$, then m' is received and the rotation is resumed until the special symbol $\sharp$ is obtained at the head of c' in which case it is received. When the above procedure has

terminated, $\mathcal{L}$ moves back to its original state and the usual execution of the system is resumed. During the rotation process, if no message belonging to the alphabet of c is found in c', or if the special symbol $\sharp$ is "lost", then the rotation continues for ever.

Now, we show that $\mathcal{L}$ has a computation which visits s infinitely often if and only if $\mathcal{L}'$ has a computation which visits s infinitely often.

[only if] Suppose that $\mathcal{L}$ has a computation π which visits s infinitely often. We construct a computation π' of $\mathcal{L}'$ which visits s infinitely often. Each transition in π which is either a send operation, or a receive operation where the received message is at the head of c', or an observable interaction, or a message loss, is simulated in π' by executing the corresponding transition in $\mathcal{L}'$. To simulate π receiving a message m on (say) channel c where m is not at the head of c', π' performs the rotation operation without losing any messages. It is clear that m is the first message in c' belonging to the alphabet of c. This means that m is eventually received in π', and the content of c' will still be a merge of the contents of c and d.

[if] Suppose that $\mathcal{L}'$ has a computation π' which visits s infinitely often. We construct a computation π of $\mathcal{L}$ which visits s infinitely often. We observe that each time the rotation operation is invoked in a control state t, to receive a message m from (say) channel c, π' must eventually return to t. Otherwise π' would not visit s infinitely often. This means that m must be the first message in c' belonging to the alphabet of c. Hence the rotation operation can be simulated in π by first simulating all the message losses during the operation, and then executing the single transition which receives m form c. Computation steps in π' which do not belong to the rotation operation are simulated in π by executing the corresponding transitions in $\mathcal{L}$. □

5 Temporal Logic

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)*.

5.1 Propositional Linear Time Logic

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 $f : S \rightarrow 2^{Prop}$, 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:

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

We define $\Diamond\phi = \mathit{true}\ \mathcal{U}\ \phi$, 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 $f, \pi \models \phi$.

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 $\Diamond\Box\neg p$. $\Box$

5.2 Hierarchy of Decidable Problems in PTL

Relating to the hierarchy of properties expressible in PTL defined by Manna and Pnueli [MP92], the following can be concluded about lossy channel systems. Safety properties ($\Box p$), eventuality properties or guarantee properties ($\Diamond p$) and obligation properties ($\Box p \ \vee \ \Diamond q$) are decidable . Response properties ($\Box\Diamond p$), persistence properties ($\Diamond\Box p$), and reactivity properties ($\Box\Diamond p \ \vee \ \Diamond\Box q$) are undecidable. The decidability results follow from Theorem 3.1, while the undecidability results follow Corollary 4.7.

5.3 Computation tree Logic

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 $AF(AG(\neg p))$. □

6 Eventuality Properties with Fair Channels

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. In the literature, there are many different variants of fairness properties [Fra86]. In this section, we first consider the fairness property which requires that if a message is infinitely often sent to a channel, then it is infinitely often received from the channel. We study the eventuality properties problem (described in Section 3) again, this time being 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. 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 First Fairness Condition: Impartiality

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. That is, 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: First, we give an overview of the proof. 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 4.8) 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, (3) the only control state where ℓ is received is r, and (4) 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 all occurences of the message ℓ from the head of the channel. The construction of π' is such that it occurs infinitely often that π' reaches r where ℓ is available at the head of the channel. 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.

The details of the proof are as follows. Given a lossy channel system $\mathcal{L} = \langle S, s_0, A, \{c\}, M, \delta \rangle$, and a control state $r \in S$, we construct a lossy channel system $\mathcal{L}' = \langle S', s_0, A, \{c\}, M', \delta' \rangle$, and a control state $t \in S'$, 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. The definition of $\mathcal{L}'$ is as follows:

- $S' = S \cup \{s';\ s \in S\} \cup \{t\}$, where $t \notin S$.
- $M' = M \cup \{\ell\}$, where $\ell \notin M$.
- The set δ' is composed of six groups $\delta'_1, \delta'_2, \delta'_3, \delta'_4, , \delta'_5, \delta'_6$ of transitions.

  δ'_1 : $\langle s'_1, op, s_2 \rangle$, for each s_1, s_2, and op such that $\langle s_1, op, s_2 \rangle \in \delta$.

  δ'_2 : $\langle s, c!\ell, s' \rangle$, for each $s \in S$.

  δ'_3 : $\langle s, c?m, s \rangle$, for each $s \in S$ and $m \in M$.

  δ'_4 : $\langle s, c!\ell, s \rangle$, for each $s \in S$.

  δ'_5 : $\langle r', c?\ell, r' \rangle$.

  δ'_6 : $\langle s', \tau, t \rangle$, for each $s \in S$.

This means (Fig. 2 and Fig. 3) that each control state s in $\mathcal{L}$ is "split" into two control states s and s' in $\mathcal{L}'$. All incoming transitions to s in $\mathcal{L}$ still lead to s in $\mathcal{L}'$, while all outgoing transitions from s in $\mathcal{L}$ depart from s' in $\mathcal{L}'$ (according to δ'_1). Furthermore, a number of transitions are added to s and s' according to $\delta'_2, \ldots, \delta'_6$.

We represent global states of $\mathcal{L}$ and $\mathcal{L}'$ as pairs, where the first element of the pair is the control state and the second element of the pair is the content of the channel. Now we show 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.

[only if] Let $\pi = \gamma_1\ \gamma_2\ \gamma_3\ \ldots$ be a computation of $\mathcal{L}$ which visits r infinitely often. Without loss of generality we assume that message losses occur in π only when the message is at the head of c. Let γ_i be of the form $\langle s_i, x_i \rangle$. We shall construct an impartial computation π' of $\mathcal{L}'$ which does not reach t. There are two cases.

1. If there are infinitely many steps in π which are message losses or message receptions. This means that there are infinitely many i such that $x_i = m \bullet x_{i+1}$, for some $m \in M$.

We get π' from π by replacing each γ_i by a sequence of global states of $\mathcal{L}'$. Intuitively, π' simulates all the operations of π on channel c, replacing message losses by message receptions, and it performs some additional operations on the new alphabet symbol ℓ. The steps of π' are defined by the following five cases.

1. If the next step of π is that of losing a message m, then π' performs the following. (1) it sends ℓ to c using the transitions in δ'_4, (2) it loses all occurrences (if any) of ℓ at the head of c, and (3) it receives m using the transitions in δ'_3. This means that no messages belonging to M are lost in π'.

2. If π has reached a control state s, where $s \neq r$, and if the next step of π is that of receiving a message m, then π' performs the following. (1) it sends ℓ to c, using the transitions in δ'_2, (2) it loses all occurrences (if any) of ℓ at the head of c, and (3) it performs the next operation of π, using the transitions in δ'_1.

3. If π has reached a control state s, where $s \neq r$, and if the next step of π is neither losing or receiving a message, then π' performs the following. (1) it sends ℓ to c, using the transitions in δ'_2, and (2) it performs the next operation of π, using the transitions in δ'_1.

4. If π has reached the control state r, and if the next step of π is that of receiving a message m, then π' performs the following. (1) it sends ℓ to c, using the transitions in δ'_2, (2) it receives all occurrences (if any) of ℓ at the head of c, and (3) it performs the next operation of π, using the transitions in δ'_1.

5. If π has reached the control state r, and if the next step of π is neither losing or receiving a message, then π' performs the same steps as in 4.

The computation π' is impartial with respect to each $m \in M$, since there are infinitely many steps in π where messages are removed from c, and each message sent to c in π must be removed at some later stage of π. Furthermore, all removals of messages belonging to M in π' are achieved through message receptions, as each loss of a message $m \in M$ from c in a control state s is replaced by the transition $s \xrightarrow{c?m} s$ in $\mathcal{L}'$.

Now we show that π' is also impartial with respect to ℓ. We observe that whenever we send a message $m \in M$ to c in π', it is preceded by sending ℓ to c. We also note that the only occasion when removing the occurrences of ℓ in the head of c is not followed by removing the next message $m \in M$ is when performing the steps in 5. This means that after the execution of the

steps in 1, 2, or 4, we always get ℓ at the head of c. This continues to be the case at least until the next time the steps in 5 are executed. Since there are infinitely many steps in π which are message losses or message receptions, there are infinitely many times where the steps in 1,2, or 4 are executed. It follows that it occurs infinitely often that we perform the steps in 4 or 5 with ℓ available at the head of the channel, and consequently it occurs infinitely often that ℓ is received form c.

2. If there are only finitely many steps which are message losses or message receptions. This means that there is a j such that for each i with $i > j$, there is no $m \in M$ where $x_i = m \bullet x_{i+1}$. In a similar manner to case **1**, we get π' from π by replacing each γ_i by a sequence of global states of $\mathcal{L}'$. Intuitively, up to the j^{th} step, π' simulates the steps of π and performs additional operations on ℓ, in exactly the same manner as described in **1** above. After the j^{th} step, π' will in addition receive the first message (if available) of channel c using the transitions in δ'_3 and δ'_5. Notice that it will still be possible to simulate the steps of π, since no message receptions occur in π after the j^{th} step.

The computation π' is impartial since each message sent to c is eventually received.

[if] Suppose that $\mathcal{L}'$ has an impartial computation π' which does not reach t. We shall construct a computation π of $\mathcal{L}$ which visits r infinitely often. We derive π form π' by performing the following two operations on π'. (1) we skip all steps of π' which are operations on the message ℓ, and (2) we replace each step of π' which is a reception of a message m, via a transition in the set δ'_5, by losing m in π. Now, we show that π' visits r infinitely often. Due to the transitions in δ'_2 and δ'_6, deadlock cannot occur in π'. This means that the message ℓ is sent infinitely often to c during π'. It follows that π' visits r infinitely often, since π' is impartial and r is the only control state where ℓ can be received from c. It is easy to see that this implies that also π visits r infinitely often. □

6.2 Other Fairness Conditions

We define two new fairness conditions on the channels. 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 π 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. The computation π is said to be *message fair* if it is message fair with respect to all messges in M and all channels in 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. The computation π is said to be *channel fair* if it is channel fair with respect to all channels in 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. Referring to the hierarchy of temporal properties defined in [MP92], we can say that safety properties, eventuality properties (guarantee properties) and obligation properties are decidable, while response properties, persistence properties, and reactivity properties are undecidable for lossy channel systems

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. Extended Abstract.

[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.

[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.

[Fra86] N. Francez. *Fairness*. Springer Verlag, 1986.

[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.

[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.

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

[LY93] K. Larsen and W. Yi. Time-abstracted bisimulation: Implicit specification and decidability. In *Mathematical Foundations of Programming Semantics*, Lecture Notes in Computer Science, New Orleans, April 1993. Springer Verlag.

[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.

[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.

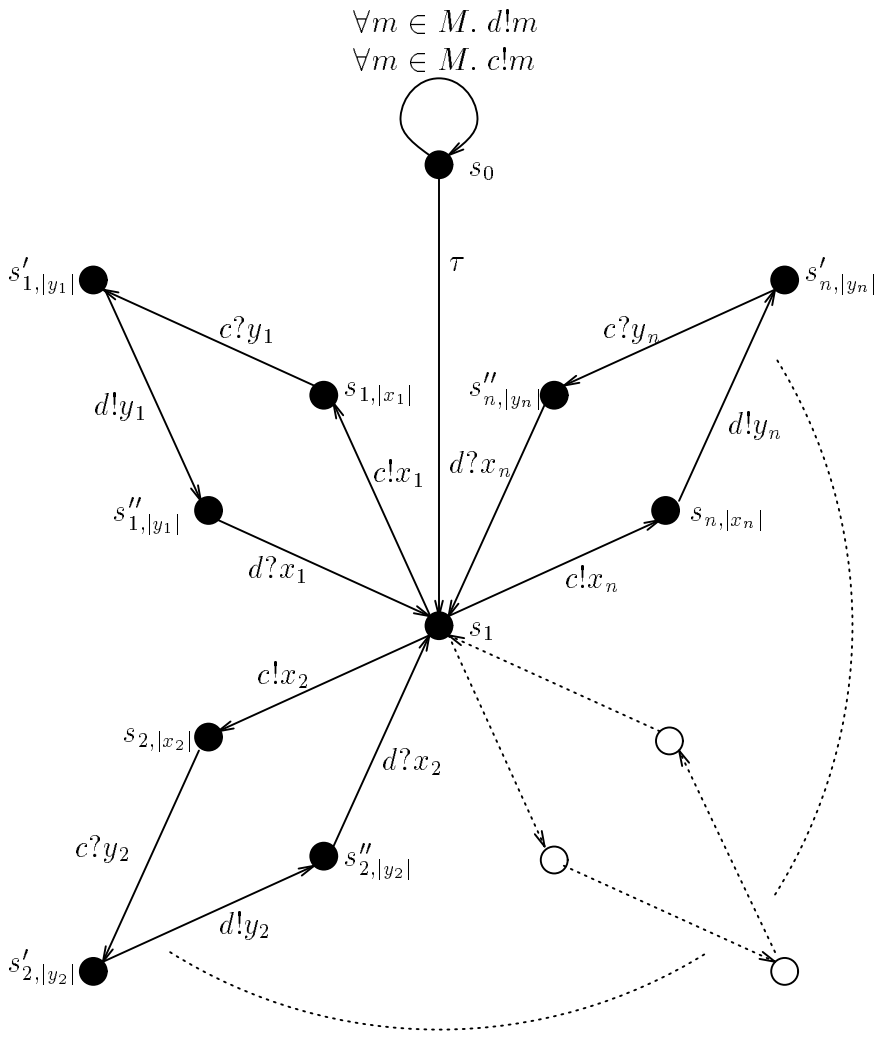

Figure 1: The lossy channel system construction for the proof of undecidability of RSP.

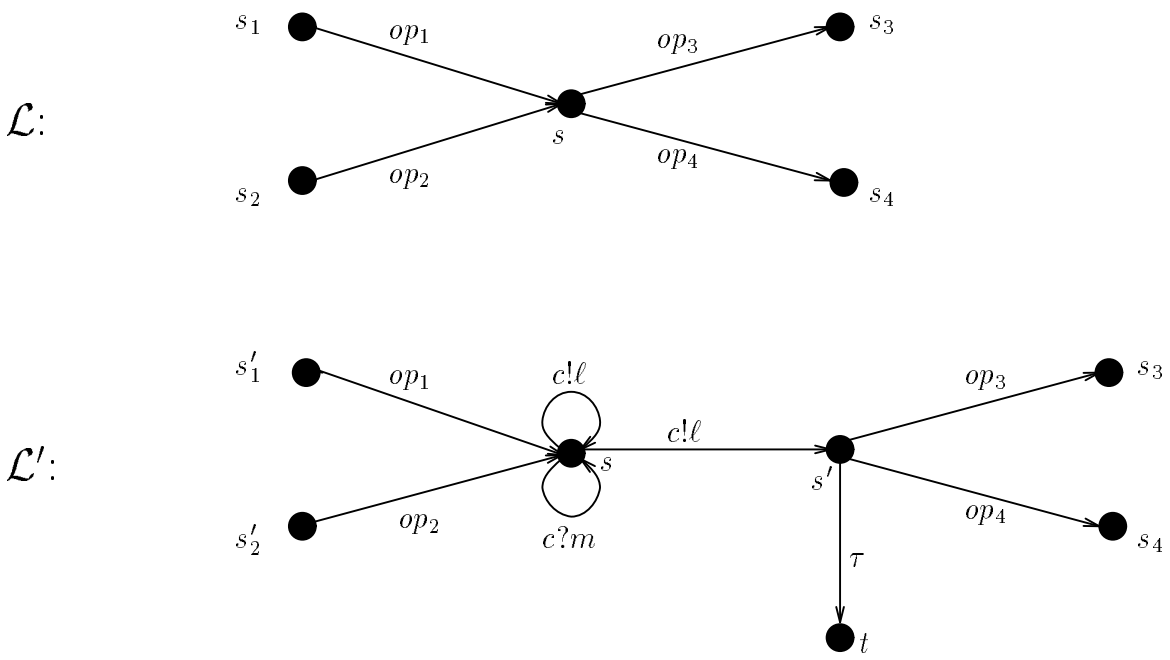

Figure 2: Splitting a control state s, where $s \neq r$, into two control states s and s'. All incoming transitions to s in $\mathcal{L}$ still lead to s in $\mathcal{L}'$, while all outgoing transitions from s in $\mathcal{L}$ depart from s' in $\mathcal{L}'$ (according to δ'_1). Also, a number of transitions are added to s and s' (according to $\delta'_2, \ldots, \delta'_6$).

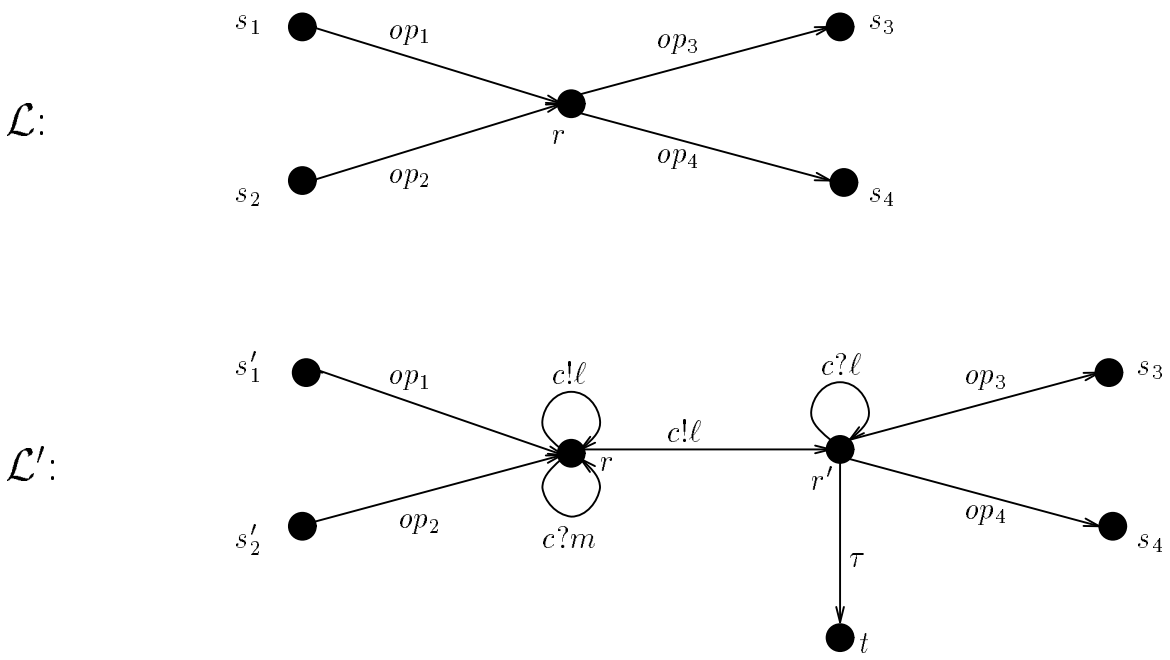

Figure 3: Splitting the control state r into two control states r and r'. All incoming transitions to r in $\mathcal{L}$ still lead to r in $\mathcal{L}'$, while all outgoing transitions from s in $\mathcal{L}$ depart from s' in $\mathcal{L}'$ (according to δ'_1). Also, a number of transitions are added to r and r' (according to $\delta'_2, \ldots, \delta'_6$).