INFORMATION AND COMPUTATION **124**, 20–31 (1996)
Article No. 0003

1996

Unreliable Channels Are Easier to Verify Than Perfect Channels

GÉRARD CÉCÉ AND ALAIN FINKEL

LIFAC, ENS de Cachan, 61 avenue du Président Wilson, 94235 Cachan Cedex, France
E-mail: {cece, finkel}@lifac.ens-cachan.fr

AND

S. PURUSHOTHAMAN IYER

Department of Computer Science, North Carolina State University, Raleigh, North Carolina 27695-8206
E-mail: purush@csc.ncsu.edu

We consider the problem of verifying correctness of finite state machines that communicate with each other over unbounded FIFO channels that are unreliable. Various problems of interest in verification of FIFO channels that can lose messages have been considered by Finkel and by Abdulla and Jonsson. We consider, in this paper, other possible unreliable behaviors of communication channels, viz., (a) duplication and (b) insertion errors. Furthermore, we also consider various combinations of duplication, insertion, and lossiness errors. Finite state machines that communicate over unbounded FIFO buffers are a model of computation that forms the backbone of the ISO standard protocol specification languages Estelle and SDL. While the assumption of a perfect communication medium is reasonable at the higher levels of the OSI protocol stack, the lower levels have to deal with an unreliable communication medium; hence our motivation for the present work. The verification problems that are of interest are *reachability*, *unboundedness*, *deadlock*, and *model-checking against CTL**. All of these problems are undecidable for machines communicating over reliable unbounded FIFO channels. So it is perhaps surprising that some of these problems become decidable when unreliable channels are modeled. The contributions of this paper are (a) an investigation of solutions to these problems for machines with insertion errors, duplication errors, or a combination of duplication, insertion, and lossiness errors, and (b) a comparison of the relative expressive power of the various errors.

1. INTRODUCTION

Finite state machines which communicate over unbounded channels have been used as an abstract model of computation for reasoning about communication protocols [5, 12] and form the backbone of the ISO protocol specification languages Estelle [8] and SDL [7]. Ever since the publication of the Alternating bit protocol [3] (the first computer communication protocol) it has been customary to assume, while modeling a protocol, that the communication channels between the processes are free of errors. Possible errors in the communication channels are treated separately, or are completely ignored. In [10] Finkel considered a model of errors, called *completely specified protocols*, in which messages from the front of a queue can be lost. He showed that the *termination* problem is solvable for this class. In [1, 2] Abdulla and Jonsson consider a slightly more general notion of message lossiness: they assume that messages from anywhere in the queue can be lost. They considered the reachability problem [1] and the model-checking problem [2] against specifications in the branching time temporal logic *CTL** [9]. They show that the reachability problem is decidable and that the model-checking problem is undecidable. This is in sharp contrast to finite state machines communicating over perfect channels, which are equivalent to Turing machines [6].

In this paper we consider two other sources of errors in unreliable channels: duplication and arbitrary insertion of messages. We show that duplication of messages, waiting in the queue to be delivered, does not decrease the power of these communicating finite state machines; they indeed are equivalent to Turing machines. On the other hand, in the case of communicating finite state machines that have arbitrary insertion errors, we show that the reachability state space of such machines is recognizable. Furthermore, we show how to compute a description of this recognizable set. Based on this description, problems such as reachability and boundedness are immediately shown to be decidable. We also consider machines that can have a combination of these errors. The contributions of this paper are:

1. Some new results on completely specified protocols, lossy systems and connections between the two.

2. A complementary view, on errors, to what has already been proposed in the literature. In particular we consider insertion errors, duplication errors, and combination of the three (insertion, duplication and lossiness) types of errors.

3. A comparison of the relative expressive power of these errors. Our finding are that insertion errors decrease

0890-5401/96 $12.00

the expressive power of the communication finite state machines the most, followed by lossiness; in sharp contrast we find that the duplication errors do not decrease the power of communicating finite state machines.

The presentation is structured as follows: In Section 2 we recall the necessary mathematical definitions. In Section 3 we present an overview of the past work on lossy channels and provide some new results. We present our results for arbitrary insertion errors in Section 4, our results for duplication in Section 5 and the various combination in Section 6. In the conclusion we offer a comparison of the relative expressive power of these errors.

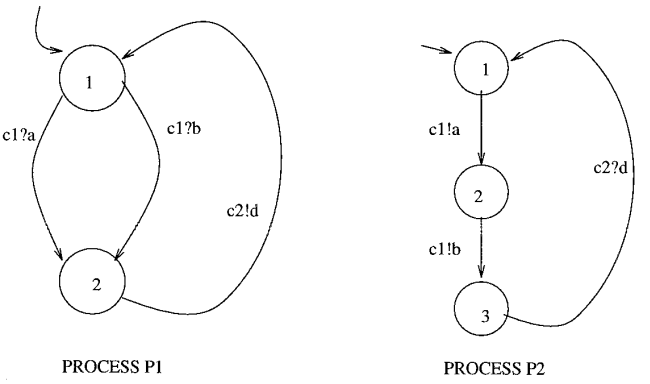


FIG. 1. Two communicating machines.

2. DEFINITIONS AND PRELIMINARIES

2.1. The Model of Communicating Finite State Machines

While it is customary to think of a network of communicating finite state machines as made up of a set of finite state processes,[1] we will talk about a single finite state control (which could be a product of the component machines) acting on a set of channels. Consider, for example, the two machines P_1 and P_2 communicating over channels c_1 and c_2, shown in Fig. 1. We could as well consider the single machine acting on two channels c_1, and c_2, as shown in Fig. 2, instead. This is possible as this single machine contains a shuffle of the actions of the two machines. It has been shown that this model is as powerful as a Turing machine [6]. Formally, we have

DEFINITION 1. A machine is a tuple $M=(S, C, \bigcup_{c\in C}\Sigma_c, s_0, \delta)$, where S is a finite set of states, $C=\{c_1, ..., c_n\}$ is a finite set of channels, Σ_c is the alphabet of channel $c\in C$, $s_0\in S$ is the distinguished initial state, and

$$\delta \subseteq S\times\left(\bigcup_{c\in C}\{c?a, c!a \mid a\in\Sigma_c\}\right)\times S$$

is the translation relation.

Notations 1.

- We will use $x\cdot y$ to emphasize the concatenation of strings x and y.
- A *global state* of M is a tuple $u=\langle s, x_1, ..., x_n\rangle$ with $s\in S$ and $x_i\in\Sigma_i^*$. Let $G(M)$ be the *set of all global states* of M.
- $u_0=\langle s_0, \varepsilon, ..., \varepsilon\rangle$ is called the *initial global state.*
- Let Σ_C be the set $\bigcup_{c\in C}\Sigma_c$.

In the following $c!a$ denotes the act of sending a message a on channel c and $c?a$ denotes the act of receiving the message a from channel c. As we are dealing with asynchronous communication any message that has been sent will be queued in the buffer to be picked up later by the receiver. The queue itself has a FIFO behavior.

DEFINITION 2. Given a machine $M=(S, C, \Sigma_C, ..., s_0, \delta)$, the *set of reachable states* of the machine M is the least set $R(M)\subseteq G(M)$ defined inductively by the following rules:

- (*initial state axiom*) $u_0\in R(M)$ is the initial state.
- If $u=\langle s, x_1, ..., x_n\rangle\in R(M)$ then

— (*output rule*) $u'=\langle s', x_1, ..., x_{i-1}, x_i\cdot a, x_{i+1}, ..., x_n\rangle\in R(M)$ provided $(s, c_i!a, s')\in\delta$. This defines the semantics of an output action. Furthermore, we will write $u\xrightarrow{(s, c_i!a, s')}u'$ to denote that u' is a *successor* of u due to the transition $(s, c_i!a, s')$.

— (*input rule*) $u''=\langle s', x_1, ..., x_{i-1}, x_i', x_{i+1}, ..., x_n\rangle\in R(M)$ provided $(s, c_i?a, s')\in\delta$ and $x_i=a\cdot x_i'$. This defines the semantics of an input action. Furthermore, we will write $u\xrightarrow{(s, c_i?a, s')}u''$ to denote that u'' is a *successor* of u due to the transition $(s, c_i?a, s')$.

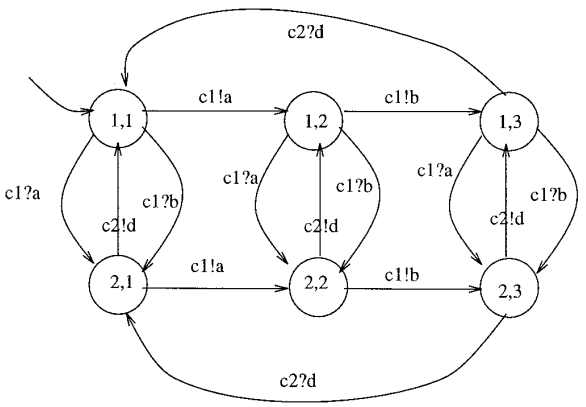


FIG. 2. Alternative view: One machine acting on two buffers.

[1] We use the words process and machine synonymously.

Notations 2. We will write $u \to u'$ whenever we do not care about the transition t which makes the machine change its global state from u to u'. As usual, we extend relation $\to$ to its transitive closure $\xrightarrow{+}$ and to its reflexive and transitive closure $\xrightarrow{*}$.

In the future we will call machines which do not model error *normal machines*.

DEFINITION 3. Given a normal machine $M = (S, C, \Sigma_C, s_0, \delta)$, an execution sequence σ is

- a finite sequence of the form $\sigma = u_1, t_1, u_2, t_2, ..., t_{n-1}, u_n$, where

$$\forall i, 1 \leqslant i \leqslant n-1, u_i \xrightarrow{t_i} u_{i+1},$$

or

- an infinite sequence $\sigma = u_1, t_1, u_2, ...$ such that

$$\forall i \geqslant 1: u_i \xrightarrow{t_i} u_{i+1}.$$

The set of words that are in the buffers can be identified with various languages classes. To that end we define the notion of *channel languages* as follows:

DEFINITION 4. Given a machine M with a set of states S and the reachability set $R(M)$, define the *channel language* of a state $s \in S$ to be

$$L(s) = \{\langle x_1, ..., x_n\rangle \in \Sigma_1^* \times \cdots \times \Sigma_n^* \mid \langle s, x_1, ..., x_n\rangle \in R(M)\}.$$

The set $R(M)$ captures the semantics of a machine M when it is acting normally, without any errors. Given a machine M, we would like to study various properties of this machine.

DEFINITION 5. Given a machine M with the reachability set $R(M)$, the following properties are of importance:

Reachability: Does a particular state $\langle s, x_1, ..., x_n\rangle$ belong to $R(M)$?

Deadlock: Does a state $u = \langle s, \varepsilon, ..., \varepsilon\rangle$ belong to $R(M)$ such that there are no successors for u?

Boundedness: Is the set $R(M)$ finite?

Finite termination: Are all of the execution sequences of machine M starting from u_0 finite?

Computation of $R(M)$: There are two questions here. The first question is: is the set $R(M)$ recognizable? If so, then the second question is: is there an algorithm which when given M would construct a finite state machine for $R(M)$?

*Model-Checking against CTL**: Given a *CTL** formula ϕ over an appropriate set of atomic propositions, does the initial state satisfy the formula ϕ?

If a normal machine has at least one FIFO queue it can be used to simulate a Turing machine. Consequently, we have:

THEOREM 1 (Brand and Zafiropulo [6]). *All of the problems of interest are undecidable for normal machines.*

2.2. Subword Ordering

Our technical treatment of unreliable channels critically depends upon the notion of the subword relation and its properties, which we now recall.

DEFINITION 6. Let Σ be a finite alphabet and $x, y \in \Sigma^*$.

- $x \preceq y$ (i.e., x is a subword of y) provided $x = a_1 \cdots a_n$ and $y = y_0 a_1 y_1 \cdots a_n y_n$, where $y_i \in \Sigma^*$ and $a_i \in \Sigma$.
- $closure(x) = \{z \in \Sigma^* \mid x \preceq z\}$.

The relation $\preceq$ is a reflexive and a transitive relation. Furthermore, it has the following property (due to Higman):

THEOREM 2 (Higman [13, 14]). *If a set of words X consists of mutually incomparable elements according to $\preceq$ then W is finite.*

Let M be a machine. We can, without loss of generality, also use $\preceq$ for global states of M: define that $\langle s, x_1, ..., x_n\rangle \preceq \langle s', x'_1, ..., x'_n\rangle$ provided $s = s'$ and for every i, $1 \leqslant i \leqslant n$, we have $x_i \preceq x'_i$. Note that Higman's lemma is still true for this extension of $\preceq$. Given a set $W \subseteq G(M)$, we will say that W is upward closed if for each $w \in W$ every element w' such that $w \preceq w'$ is also in W. Formally,

DEFINITION 7. Let M be a machine and $W \subseteq G(M)$. Define $closure(W) = \{w' \in G(M) \mid \exists w \in W,\ w \preceq w'\}$. Consequently, a set of global states W is said to be upward closed if

$$W = closure(W)$$

Consider the minimal elements of an upward closed set. Since any two minimal elements are mutually incomparable, we have, by Higman's lemma, that the set $min(W)$ of minimal elements of any upward closed set W is necessarily finite. So we have

LEMMA 1. *Every upward closed set $W \subseteq G(M)$ is recognizable. Indeed, we have*

$$W = \bigcup_{\langle s, x_1, ..., x_n\rangle \in min(W)} \langle x, X_1, ..., X_n\rangle$$

with $X_i = closure(x_i)$.

Proof. As $min(W)$ is finite, W is a finite union of product of recognizable languages. Indeed, if $x_i = a_1 \cdots a_m$ then $closure(x_i) = \Sigma_i^* a_1 \Sigma_i^* a_2 \Sigma_i^* \cdots \Sigma_i^* a_m \Sigma_i^*$, which is trivially a recognizable language. Thus W is also recognizable (see [4]). ∎

A second consequence of Higman's lemma is that there is no infinite sequence of successively strictly larger upward closed sets. Formally, we have:

LEMMA 2. *For each sequence* $\{W_i\}_{i \geqslant 0}$ *of successively larger upward closed set (i.e.* $W_i \subseteq W_{i+1}$*), there exists a finite* l *such that* $W_l = \bigcup_{i \geqslant 0} W_i$ *(which implies that* $\forall k \geqslant l$, $W_k = W_l$*).*

Proof. As all W_i are upward closed, so also is $W = \bigcup_{i \geqslant 0} W_i$. From Higman's lemma, $min(W)$ is finite. Thus there exists a finite l such that $min(W) \subseteq W_l$. As each element of $min(W)$ belongs to a W_i, it suffices to take W_l as the biggest of these chosen W_i. We then have $W = closure(min(W)) \subseteq closure(W_l) = W_l$. And thus $W_l = W = \bigcup_{i \geqslant 0} W_i$. ∎

3. RESULTS ON LOSSY MACHINES

In this section we will recall existing results from the literature so that we can compare the various kinds of errors in an uniform framework. In this process we will also show some new results. In the literature two models, *completely specified protocols* [10] and *machines capable of lossiness errors* [1], have been considered. They differ in that completely specified protocols can only lose messages from the front of the queue, whereas the machines capable of lossiness errors can lose messages anywhere from the queue. Though the two models are not equivalent (in that their reachability sets are not the same), they are related. In [11], the authors consider specifications of two faulty queues, the first modeling the queue of a completely specified protocol and the second modeling the queue of a machine capable of lossiness errors. They show that the two specifications are observationally equivalent but not bisimilar.

We now recall the definitions:

DEFINITION 8. A *completely specified protocol* (or front lossy system) is a machine $M = (S, C, \Sigma_C, s_0, \delta)$ whose reachability set is the least set $R_{csp}(M) \subseteq G(M)$ defined by the following rules:

- The initial state axiom, output rule, and input rule are as for a normal machine.
- $\forall i,\ 1 \geqslant i \geqslant n$, and $\forall a \in \Sigma_i$, if $u = \langle s, x_1, ..., x_{i-1}, ax_i, x_{i+1}, ..., x_n \rangle \in R_{csp}(M)$, then $u' = \langle s, x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n \rangle \in R_{csp}(M)$.

We write $u \xrightarrow{L_{csp}} u'$ to denote that u' is a successor of u caused by the message loss transition.

The notion of an execution sequence for normal machines carries over to completely specified protocols. Finkel showed the following:

THEOREM 3 (Finkel [10]). *The finite termination problem for completely specified protocols is solvable.*

Let us now consider machines that are capable of lossiness errors.

DEFINITION 9. A machine that is *capable of lossiness errors* is a machine $M = (S, C, \Sigma_C, s_0, \delta)$ whose reachability set is the least set $R_L(M) \subseteq G(M)$ defined by the following inductive rules:

- The initial state axiom, input rule, and output rule are as for normal machines.
- $\forall 1 \leqslant i \leqslant n$ and $\forall a \in \Sigma_i$, if $u = \langle s, x_1, ..., x_{i-1}, x_i' a x_i'', x_{i+1}, ..., x_n \rangle \in R_L(M)$, then $u' = \langle s, x_1, ..., x_{i-1}, x_i' x_i'', x_{i+1}, ..., x_n \rangle \in R_L(M)$.

We will write $u \xrightarrow{L} u'$ to denote that u' is a successor of u caused by the message loss transition.

The notion of execution sequences carries over from the definition of completely specified protocol. Abdulla and Jonsson show the following:

THEOREM 4 (Abdulla and Jonsson [1, 2]). *The following are true for machines capable of lossiness errors*:

1. *The reachability set is recognizable.*
2. *The finite termination problem is solvable.*
3. *The reachability problem and the deadlock problems are decidable.*
4. *The model-checking problem is undecidable.*

For the sake of completeness (and as our proofs depend upon some of these details) we will briefly explain the proofs of the above mentioned items. Note that the complement $\overline{R_L(M)}$ of the reachability set is upward closed. Thus $\overline{R_L(M)}$ is recognizable (Lemma 1). This implies that $R_L(M)$ is also recognizable since recognizable sets are closed under complementation (see [4]).

A consequence of Higman's lemma is the following (cf. [14]). For every infinite sequence $u_1, u_2, ...$ of global states of a machine M, there exist two states u_i and u_j such that $i < j$ and $u_i \preceq u_j$. From this lemma and the fact that $(u_i \preceq u_j \Rightarrow u_j \xrightarrow{*} u_i)$, there exist a reachable global state greater than one of its ancestors if and only if there exists an infinite execution sequence.

The third result was proven by the construction of a partial backward relation $\rightharpoonup$ between global states. Abdulla and Jonsson show that $u_0 \xrightarrow{*} u \Leftrightarrow u \overset{*}{\rightharpoonup} u_0$. Furthermore, by Higman's lemma, they show that backward reachability can be checked. In the next theorem, we give a shorter proof of the decidability of the reachability problem.

THEOREM 5. *Given a machine M and a subset $\Gamma \subseteq G(M)$, the set $\mathscr{P}(\Gamma) = \{u \in G(M)\} \mid \exists u' \in \Gamma, u \xrightarrow{*} u'\}$ of all predecessors of Γ is recognizable. Furthermore, a finite state machine description of $\mathscr{P}(\Gamma)$ is computable if a finite state machine description of closure (Γ) is computable.*

Proof. Let us define the family of sets $\{P_i\}_{i \geqslant 0}$ such that

$$P_0 = \Gamma$$
$$P_{i+1} = P_i \cup \{u \in G(M) \mid \exists u' \in P_i, u \to u'\}.$$

We have $\mathscr{P}(\Gamma) = \bigcup_{i \geqslant 0} P_i$. By the second item of Definition 9, $\mathscr{P}(\Gamma)$ is an upward closed set because: $u_1 \preceq u_2$, $u_1 \xrightarrow{*} u \Rightarrow u_2 \xrightarrow{*} u_1 \xrightarrow{*} u$. Thus $\mathscr{P}(\Gamma)$ is recognizable (Lemma 1).

To show that a finite state machine description of $\mathscr{P}(\Gamma)$ can be computed if a finite state machine description of *closure*(Γ) is computable we use the alternative characterization: $\mathscr{P}(\Gamma) = \bigcup_{i \geqslant 0} P'_i$, where

$$P'_0 = closure(\Gamma)$$
$$P'_{i+1} = closure(P'_i \cup \{u \in G(M) \mid \exists u' \in P'_i, u \to u'\}).$$

Note that in the definition of P'_{i+1} it is the one-step reachability relation, and not its transitive closure, which is used. Thus, if *closure*(Γ) is computable, then each of the P'_{i+1} is easily computable from P'_i. Since $\{P'_i\}_{i \geqslant 0}$ is a sequence of successively larger upward closed sets, we have, from Lemma 2 that there exists an l such that $\mathscr{P}(\Gamma) = P'_l$. Consequently, $\mathscr{P}(\Gamma)$ is computable. ∎

Note. The decidability part of Theorem 5 can be easily seen to hold when Γ is finite or recognizable.

COROLLARY 1. *The reachability problem is decidable for a machine M capable of lossiness errors.*

Proof. Let u be the global state of M we want to check for reachability. We have

$$u_0 \xrightarrow{*} u \Leftrightarrow u_0 \in \mathscr{P}(\{u\}).$$

The second part is decidable by the above theorem. ∎

The fourth result of Theorem 4—undecidability of model-checking lossy machines against specifications in *CTL** formulae—is based on the undecidability of the *recurrent state problem* (RSP). For a class of machines $\mathscr{C}$, RSP can be stated as follows:

> Given a machine M of a class $\mathscr{C}$, and given a state s of M, is there an infinite execution sequence starting from u_0 that visits s infinitely often?

We have the following:

THEOREM 6 (Abdulla and Jonsson [2]). *The RSP is undecidable for machines capable of lossiness errors.*

COROLLARY 2 (Abdulla and Jonsson [2]). *The model-checking for a lossy system against CTL* specifications is unsolvable.*

Since we will use the same argument for completely specified protocols, let us recall the proof (see [2]). If the model-checking problem were solvable it would be possible to solve the RSP problem by checking whether a given machine M has the property $E(G(F(@s)))$—which posits the existence of an execution sequence starting from u_0 in which it is always true that eventually control would be at state s.

The reachability set $R_L(M)$ of a machine capable of lossiness errors is recognizable. Furthermore, since $\overline{R_L(M)}$ is upward closed, $min(\overline{R_L(M)})$ is finite and gives a representation of $\overline{R_L(M)}$ and thus of $R_L(M)$. One could think it possible to compute $min(\overline{R_L(M)})$ in order to deduce $R_L(M)$. Unfortunately, we will now prove that there is no algorithm for computing the reachability set of a machine capable of lossiness.

THEOREM 7. *For the class of machines which are capable of losing messages there exists no procedure which when given a machine M would compute a finite state machine representation of $R_L(M)$.*

Proof. The proof of the theorem follows from the following claim:

> For the class of machines capable of lossiness errors there is no algorithm which, when given a machine M and a state s of M, can decide whether $L(s)$ is a finite set or not.

To prove the claim, given a machine M we can construct a new machine M' which is similar to M but has a new channel c_n whose alphabet is $\#$. Also add a new state s_n to M'. Furthermore, all transitions of the form (s', z, s) in M are replaced by two transitions: (s', z, s_n) and $(s_n, c_n!\#, s)$ in M'. Now, clearly, the channel c_n will grow unboundedly *if and only if* the state s is visited infinitely often. Consequently, by Theorem 6, our claim is valid.

Now the undecidability of computing the finiteness of $L(s)$ can be reduced, trivially, to the undecidability of computing the channel language of s for the machine M. ∎

In the following we set up links between completely specified protocols and machines capable of lossiness errors.

LEMMA 3. *Let M be a machine, let $\to_L$ be the transition relation between global states of M considered as a lossy machine, and $\to_{csp}$ the transition relation between global*

states of M considered as a completely specified protocol. Let u, u', and u'' be global states such that $u \preceq u'$ and $u \to_L u''$; then there exists a global state u''' such that $u' \xrightarrow{+}_{\text{csp}} u'''$ and $u'' \preceq u'''$.

Proof. By an examination of the different cases of $u \to_L u''$. ▮

THEOREM 8. *Given a machine M we have the following results.*

1. *The finite termination problem is decidable for completely specified protocols.*

2. *$R_L(M)$ and $R_{\text{csp}}(M)$ have the same deadlock states. The set of all deadlock states of $R_{\text{csp}}(M)$ is computable.*

3. *The model-checking problem is undecidable for completely specified protocols.*

4. *$R_L(M)$ is finite iff $R_{\text{csp}}(M)$ is finite.*

Proof. Let M_L stand for M considered as a lossy machine, and M_{csp} for M considered as a completely specified protocol. Note that the first item has been proved in [10] for which we provide a shorter proof here.

1. The proof follows from the proof for the corresponding statement for the class of lossy machines. Clearly, an execution sequence of M_{csp} is also an execution sequence of M_L. Thus, if there is an infinite execution sequence in M_{csp} there is also an infinite execution sequence in M_L. Conversely, by Lemma 3, if there is an infinite execution sequence in M_L then there is also an infinite execution sequence in M_{csp}. An application of Theorem 4 completes the proof.

2. We have $R_{\text{csp}}(M) \subseteq R_L(M)$. Thus a deadlock state in $R_{\text{csp}}(M)$ is also a deadlock state in $R_L(M)$. Conversely, by the use of Lemma 3, we have

$$\begin{aligned}&(u_0 \xrightarrow{*}_L \langle s, \varepsilon, ..., \varepsilon\rangle)\\ &\quad\Rightarrow (\exists u''' \in G(M), \langle s, \varepsilon, ..., \varepsilon\rangle \preceq u''', \text{ and}\\ &\qquad u_0 \xrightarrow{*}_{\text{csp}} u''' \xrightarrow{*}_{\text{csp}} \langle s, \varepsilon, ..., \varepsilon\rangle).\end{aligned}$$

Note that $\langle s, \varepsilon, ..., \varepsilon\rangle$ and u''' have the same local state. Thus a deadlock state in $R_L(M)$ is also a deadlock state in $R_{\text{csp}}(M)$. By Theorem 4 the result follows.

3. By the proof of subpart (1) the infinite execution sequences of M_{csp} and M_L are in correspondence with each other such that the control component of the corresponding global states are the same. With this observation the result follows from Corollary 2.

4. Clearly, $R_{\text{csp}}(M)$ is finite if $R_L(M)$ is finite. By the proof of subpart (2), we have

for every reachable state $u \in R_L(M)$ there is a state $u' \in R_{\text{csp}}(M)$ such that $u \preceq u'$.

Since there are a finite number of global states smaller than a given state, $R_L(M)$ is finite if $R_{\text{csp}}(M)$ is finite. ▮

The nature of the reachability set for completely specified protocols is not know yet. But if it happens to be recognizable then there is no algorithm to compute a finite state machine description of that set.

THEOREM 9. *If the class of reachability set of completely specified protocols is contained with in the class of recognizable sets then there can be no algorithm which can compute the reachability set of an arbitrary machine.*

Proof. Similar to the proof of Theorem 7, and makes use of the undecidability of RSP for completely specified protocols. ▮

4. INSERTION ERRORS

We will now define what it means for a machine to have insertion errors. While the syntax of a machine capable of insertion errors is no different from a normal machine, its semantics (i.e., its set of reachable states) is different.

DEFINITION 10. A machine $M = (S, C, \Sigma_C, s_0, \delta)$ capable of *insertion errors* has the reachability set $R_I(M) \subseteq G(M)$ which is the least set satisfying the following inductive specifications:

- The initial state axiom, output rule, and input rule are the same as for normal machines.
- $\forall 1 \leqslant i \leqslant n$ and $\forall a \in \Sigma_i$, if $u = \langle s, x_1, ..., x_{i-1}, x_i' x_i'', x_{i+1}, ..., x_n\rangle \in R_I(M)$, then $u' = \langle s, x_1, ..., x_{i-1}, x_i' a x_i'', x_{i+1}, ..., x_n\rangle \in R_I(M)$.

We will write $u \xrightarrow{I} u'$ to denote that u' is a successor of u caused by an insertion transition. ▮

The notion of an execution sequence carries over from earlier definitions.

4.1. Computation of Reachability Set for an Insertion Machine

We will now show how to compute the set $R_I(M)$ for an insertion machine. It is interesting to compare the following proof with that of Theorem 5.

THEOREM 10. *Given a machine M and a subset $\Gamma \subseteq G(M)$, the set $\mathscr{S}(\Gamma) = \{u \in G(M) \mid \exists u' \in \Gamma,\ u' \xrightarrow{*} u\}$ of all successors of Γ is recognizable. Furthermore, a finite state machine description of $\mathscr{S}(\Gamma)$ is computable provided a similar description of closure (Γ) is computable.*

Proof. Let us define the family of sets $\{S_i\}_{i \geqslant 0}$ such that

$$S_0 = \Gamma, \qquad \text{and}$$
$$S_{i+1} = S_i \cup \{u \in G(M) \mid \exists x' \in S_i, u' \to u\}.$$

We have $\mathscr{S}(\Gamma) = \bigcup_{i \geqslant 0} S_i$. By the second item of Definition 10, $\mathscr{S}(\Gamma)$ is an upward closed set because: $u_1 \preceq u_2$, $s \xrightarrow{*} u_1 \Rightarrow u \xrightarrow{*} u_1 \xrightarrow{*} u_2$. Thus $\mathscr{S}(\Gamma)$ is recognizable (Lemma 1).

We also have $\mathscr{S}(\Gamma) = \bigcup_{i \geqslant 0} S'_i$, where

$$S'_0 = closure(\Gamma), \qquad \text{and}$$
$$S'_{i+1} = closure(S'_i \cup \{u \in G(M) \mid \exists u' \in S'_i, u' \to u\}).$$

As $closure(\Gamma)$ is recognizable and computable (by assumption), each S'_{i+1} is recognizable and computable from S'_i. Furthermore, $\{S'_i\}_{i \geqslant 0}$ is a sequence of successively larger upward closed sets. Thus from Lemma 2 there exists an l such that $\mathscr{S}(\Gamma) = S'_l$. Thus, $\mathscr{S}(\Gamma)$ is computable. ▮

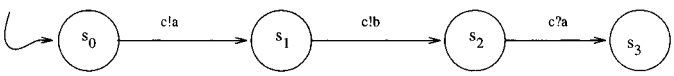
COROLLARY 3. *The reachability set $R_I(M)$ is recognizable and computable for a machine M capable of insertion errors.*

Proof. The proof follows from the observation that

$$R_I(M) = \mathscr{S}(\{u_0\})$$

and the theorem given above. ▮

It is tempting to think that the channel language for every state would be $\Sigma_1^* \times \cdots \times \Sigma_n^*$. But this is not so, as the channel language for states s_1, s_2 and s_3 of the following machine, with one channel c, does not include ε:



Since the reachability set is recognizable and computable, all of our problems of interest (deadlock, reachability, unboundedness, etc.) can immediately be solved. The reachability is reduced to the membership of a recognizable set for which we have a complete description. No state can be a deadlock state because there are always transitions out of every state. Finally, every machine is obviously unbounded as an unbounded number of messages can be inserted into the buffer. Consequently,

THEOREM 11. *We have the following for machines capable of insertion errors*:

- *The channel language is recognizable and a finite state machine description is computable.*
- *The reachability problem is decidable.*
- *The deadlock, finite termination, and boundedness problems are trivially solvable.*

Proof. The proof follows from Corollary 3. ▮

5. DUPLICATION ERRORS

In this section, we consider the problem of analyzing channels that can arbitrarily duplicate message. The formal definition of a machine capable of *duplication errors* is as follows:

DEFINITION 11. A machine $M = (S, C, \Sigma_C, s_0, \delta)$ is capable of *duplication errors* when its reachability set $R_D(M) \subseteq G(M)$ is defined as the least set satisfying the following rules:

- The initial state axiom, input rule, and output rule are the same as for normal machines.
- $\forall 1 \leqslant i \leqslant n$ and $\forall a \in \Sigma_i$, if $u = \langle s, x_1, ..., x_{i-1}, x'_i a x''_i, x_{i+1}, ..., x_n \rangle \in R_D(M)$, then $u' = \langle s, x_1, ..., x_{i-1}, x'_i a a x''_i, x_{i+1}, ..., x_n \rangle \in R_D(M)$.

We will write $u \xrightarrow{D} u'$ to denote that u' is a successor of u due to the duplication transition.

Note that the notion of execution sequences carries over from previous definitions.

We will show in the following that machines which are capable of duplication errors are as powerful as Turing machines and therefore none of the verification problems we are interested in are decidable. Without loss of generality, we will assume that there is a single channel in the machine.

The problem with our current definition of machines capable of duplication errors is that it is not possible to distinguish between a sequence of identical messages due to duplication errors and a sequence of identical messages due to the normal behavior of a machine. We can take care of this distinction as follows:

> For all normal behavior of the machine every message in the queue is to be followed by a letter (say #) not in the alphabet of the machine.

This is easily achieved by creating extra states and always sending a # after sending a normal message. More pictorially, the transformations look as follows

$$\text{each transition } t = p \xrightarrow{c!a} q \text{ is transformed to}$$
$$p \xrightarrow{c!a} r_t \xrightarrow{c!\#} q$$

and

$$p \xrightarrow{c?a} q \text{ is transformed to } p \xrightarrow{c?\#} p, \text{ and } p \xrightarrow{c?a} q,$$

where r_t are new states not in S.

We therefore need only consider machines that have the following properties:

- There is only one channel in the machine.
- For every state $s \in S$ the channel language $L(s)$ does not contain any word with two consecutive occurrences of the same letter.

Call such machines *non-duplicate machines.*

Since a normal machine is Turing powerful, the set of channel languages of a normal machine is recursively enumerable. Furthermore, as the transformation from the language of normal machines to the language of non-duplicate machines is a simple homomorphism, we infer that the channel language of non-duplicate machines are also recursively-enumerable. Let us now consider the channel languages of machines that are capable of duplication errors. Let Σ be a finite alphabet; for every non-duplicate language $L \subseteq \Sigma^*$, define

$$L_D = \{a_1^+ a_2^+ \cdots a_m^+ \mid a_1 a_2 \cdots a_m \in L\}.$$

Furthermore, define a function $f: \Sigma^* \to \Sigma^*$ as follows:

$$\begin{aligned} f(\varepsilon) &= \varepsilon \\ f(a) &= a, \quad \forall a \in \Sigma \\ f(x_1 a a x_2) &= f(x_1 a x_2), \quad \forall x_1, x_2 \in \Sigma^*, \quad a \in \Sigma \\ f(x_1 a b x_2) &= f(x_1 a)\, f(b x_2), \quad \forall x_1, x_2 \in \Sigma^*, \quad a, b \in \Sigma \\ &\quad \text{and } a \neq b \end{aligned}$$

Note that the function f merely squeezes out all repetitions with in a word. We now have the following obvious condition for a word to be in L.

LEMMA 4. *$x \in L$ iff $x \in L_D$ and $f(x) = x$.*

If L_D happens to be recursive then there is an algorithm to check whether a word x is in L_D. Clearly, it is trivial to check whether $f(x) = x$ holds for any word x. This implies that if L_D is recursive then L is also recursive. Given that L is known to be recursively enumerable, we have:

THEOREM 12. *The following statements are true*:

- *L_D is recursively enumerable.*
- *The channel language of a duplication machine is recursively enumerable.*
- *No non-trivial property of duplication machines (for instance, reachability, deadlock detection, finite termination, model-checking) is decidable.*

6. COMBINATION OF ERRORS

In this section we will consider various combinations of the three errors (a) duplication, (b) insertion, and (c) lossiness. We will establish in each of these cases whether a particular problem is decidable or not.

6.1. Lossiness and Insertion

Let M be a machine that can lose messages, and let $R_L(M)$ be the set of reachable states of M. The reachability set of such a machine is, by definition, downward closed:

$$\forall u' \preceq u \qquad \text{if} \quad u \in R_L(M) \text{ then } u' \in R_L(M).$$

Consequently, a machine that can have both lossiness and insertion errors has the following property:

LEMMA 5. *Let machine $M = (S, C, \Sigma_C, s_0, \delta)$ be a machine that is capable of having both lossiness and insertion errors. The channel language for any $s \in S$ has the following property*:

$$L(s) = \begin{cases} \Sigma_1^* \times \cdots \times \Sigma_n^* & \text{if there is a sequence} \\ & \text{of transitions from } s_0 \text{ to } s \text{ in } M \\ \varnothing & \text{otherwise.} \end{cases}$$

Given this lemma, the machines that are capable of both lossiness and insertion do not have any significant property: all states are reachable, there can be no deadlocks, and the machines are always unbounded. We thus have:

THEOREM 13. *For a machine that has both lossiness and insertion errors the reachability, finite termination, deadlock, and boundedness problems are trivially solvable.*

6.2. Duplication and Insertion

A machine that is capable of duplication and insertion does not behave any differently than a machine that is capable of only having insertion errors. This is because every duplication error is also an insertion error. Consequently our analysis for insertion errors presented in Section 5 holds here.

THEOREM 14. *We have the following for machines capable of insertion errors and duplication errors*:

- *The reachability set is recognizable and computable.*
- *The reachability problem is decidable, and the deadlock, finite termination, and unboundedness problems are trivially solvable.*

6.3. Duplication and Lossiness

Before considering both lossiness and duplication errors together, we will present another machine model that is equivalent to the machines that are capable of duplication errors. With this new model it would be easy for us to show that the class of machines capable of both duplication and lossiness errors is a subclass of machines containing only lossiness errors; consequently, all the decidability results for lossy machines [1] are applicable to machines that have both lossiness and duplication errors.

DEFINITION 12. Given a machine $M = (S, C, \Sigma_C, s_0, \delta)$, define the *duplication completion* $D(M)$ as follows:

For each $t=(s, c!a, s')\in\delta$ create a new state r_t and replace the transition t with the transitions $(s, c!a, r_t)$, $(r_t, c!a, r_t)$, and (r_t, ε, s'). The last of these do not affect any channel and are called ε-transitions.

LEMMA 6. *The reachability set of a machine M that is capable of duplicating messages is the same as the reachability set of duplication completion of M, viz. $R(D(M))$.*

Now a machine $D(M)$ is no different from a normal machine, as far as its semantics goes. Consequently, instead of considering M to be capable of duplication and lossiness errors we can consider $D(M)$ as having lossiness errors. This in turn implies that the reachability problem is decidable for the class of machines that have duplication and lossiness errors. Thus, we have

THEOREM 15. *The reachability problem for a machine that has both duplication and lossiness errors is solvable. The finite termination problem and the deadlock problem are also solvable.*

Note that the representation of a machine capable of lossiness and duplication in terms of a machine capable of lossiness does not necessarily preserve the undecidability results.

6.3.1. *Reachability Set and Model Checking*

We will show in this subsection that there is no algorithm to compute a finite state machine description of the reachability set for a machine that is capable of lossiness and duplication. We also show that the model-checking against *CTL** is undecidable for this class of processes. For the first problem we will reduce the uncomputability of the reachability set for machines capable of lossiness to the problem on hand. For the second problem our proof will consist of reducing RSP for machines capable of lossiness to the problem on hand.

Given an arbitrary machine M capable of lossiness errors, we will construct another machine M' capable of both lossiness and duplication errors such that if we can compute the reachability set of M' then we can also compute the reachability set of M. Furthermore we will show that if the model-checking question can be answered for M' then the RSP question can be answered for M. The central idea is to construct M' such that duplicate messages are of no use! Note that we will have to distinguish between a sequence of identical letter due to duplication and a sequence of identical messages due to the normal behavior of a lossy machine (for example, $a\mapsto aa$ by a duplication of a and $aba\mapsto aa$ by deletion of b). We will take care of this apparent ambiguity by sending an extra character $\#$ after each message send. Then we will consider two identical consecutive messages (i.e., not separated by $\#$) in the channel as being due to duplications.

Formally, define the function $f'\colon(\Sigma\cup\{\#\})^*\to\Sigma^*$ as $f'=g\circ f$. Function f has already been defined in Section 5, while function g is the morphism $g\colon(\Sigma\cup\{\#\})^*\to\Sigma^*$ such that $g(a)=a$ for all $a\in\Sigma$ and $g(\#)=\varepsilon$. For example, $f'(\#aaa\#bca\#\#a\#b)=abcaab$. The function f' first removes messages introduced due to duplication errors and then the message separator $\#$.

LEMMA 7. *If L is a recognizable language, defined by a finite state machine, then the language $f'(L)$ is also a recognizable language and we can compute a finite state machine describing it.*

Proof. As g is a morphism, it suffices to describe an automaton $A'=(\Sigma', Q', q'_0, F', \delta')$ defining $f(L)$ for a recognizable language L defined by an automaton $A=(\Sigma, Q, q_0, F, \delta)$. Let Σ be the finite alphabet, Q the finite set of states, $q_0\in Q$ the initial state, and F the set of final states. Assume, without loss of generality that δ is deterministic.

We now propose the components of M' as: $\Sigma'=\Sigma$, $Q'=(Q\times\Sigma)\cup\{(q_0,\bot),(q,\top)\}$, $q'_0=(q_0,\bot)$, and $F'=\{(q,a)\mid q\in F, a\in\Sigma\}\cup\{(q_0,\bot)\mid q_0\in F\}$. Furthermore, define the transition function δ' as

$$\delta'((q_0,\bot),b)=(q',b)\quad\text{iff}\quad q'=\delta(q_0,b)$$

$$\delta'((q,a),b)=\begin{cases}(q',b) & \text{if}\quad b\neq a,\ \exists u\in a^*,\ q'=\delta(\delta(q,u),b)\\ (q,\top) & \text{otherwise}\end{cases}$$

$$\delta'((q,\top),b)=(q,\top)\quad\text{for all}\quad b\in\Sigma'.$$

It should be easy to see that M' accepts only words that have no repetition, $L(M')=f'(L(M))$. ∎

Getting back to our original goal of setting up a reduction between the computation of reachable states of lossy machines and computation of reachable states of machines with lossiness and duplication, let us now define the machine $M'=(S', C, \Sigma'_C, s_0, \delta')$ associated with the machine $M=(S, C, \Sigma_C, s_0, \delta)$. Define $\forall c\in C$: $\Sigma'_c=\Sigma_c\cup\{\#\}$ and $S'=S\cup\{r_t\mid t\in\delta\}$. Define the transition function of M' as

$$\begin{aligned}&(p, c!a, r_t),(r_t, c!\#, q)\in\delta' \quad\text{iff}\quad t=(p, c!a, q)\in\delta\\ &(p, c?a, r_t),(r_t, c?\#, q)\in\delta' \quad\text{iff}\quad t=(p, c?a, q)\in\delta.\end{aligned}\tag{1}$$

Given an execution sequence $\sigma=u_1t_1u_2t_2\cdots$, define $T(\sigma)=t_1t_2\cdots$ to be the transition sequence in σ. Define $T(M)\subseteq(\delta\cup\{L\})^*\cup(\delta\cup\{L\})^\infty$ to be the set of all transition sequences obtained from the execution sequences

of M. Similarly, define $T(M') \subseteq (\delta \cup \{L, D\})^* \cup (\delta \cup \{L, D\})^\infty$ to be the set of all transition sequences obtained from the execution sequences of M'.

Notations 3. If $v = t_1 \cdots t_n$ is a sequence of transitions then we will write $u \xrightarrow{v}_M u'$ to say that there exist states $u_1, ..., u_n$ such that $u = u_1$, $\forall i\ (1 \leqslant i \leqslant n-1)\ u_i \xrightarrow{t_i}_M u_{i+1}$, and $u_n \xrightarrow{t_n}_M u'$. If $n = 0$ then we say that $u \xrightarrow{\varepsilon}_M u'$ if $u = u'$.

As we are interested in the state changes that take place in any execution sequence due to either an input or an output action, we can define two morphisms *css* (communicating states sequence): $T(M) \to S^* \cup S^\infty$ and *css'*: $T(M') \to S^* \cup S^\infty$, such that

$$css((p, c!a, q)) = css((p, c?a, q)) = p$$
$$css(L) = \varepsilon$$

and *css'* is defined similarly as

$$css'((p, c!a, r_t)) = css'((p, c?a, r_t)) = p$$
$$css'((r_t, c!\#, q)) = css'((r_t, c?\#, q)) = css'(D)$$
$$= css'(L) = \varepsilon.$$

Recall that M is only capable of lossiness errors while M' is capable of lossiness and duplication errors! Note that *css'* applied to an infinite sequence yields a finite sequence, as the result, only if that sequence contains an infinite tail of duplications.

We are now ready to consider the connection between the reachable global states (resp. the communicating states sequences) of M and the reachable global states (resp. the communicating states sequences) of M'.

LEMMA 8. 1. *If $v \in T(M)$ is a sequence of transitions such that $\langle p, x_1, ..., x_n\rangle \xrightarrow{v}_M \langle q, y_1, ..., y_n\rangle$ then there exist a sequence of transitions $v' \in T(M')$ such that $\langle p, x'_1, ..., x'_n\rangle \xrightarrow{v'}_{M'} \langle q, y'_1, ..., y'_n\rangle$ with $x'_i = a_1 \# a_2 \# \cdots \# a_m \#$ provided $x_i = a_1 \cdots a_m$, $y'_i = b_1 \# b_2 \# \cdots \# b_k \#$ provided $y_i = b_1 \cdots b_k$, and $css'(v') = css(v)$.*

2. *Let $v' \in T(M')$ be a sequence of transitions, $(x_i)_{i=1,...,n}$ and $(x'_i)_{i=1,...,n}$ be two tuples of words such that $x_i = f'(x'_i)$. If $\langle p, x'_1, ..., x'_n\rangle \xrightarrow{v'}_{M'} \langle q, y'_1, ..., y'_n\rangle$ with $p, q \in S$ then there exists a sequence of transitions $v \in T(M)$ such that $\langle p, x_1, ..., x_n\rangle \xrightarrow{v}_M \langle q, y_1, ..., y_n\rangle$ with $y_i = f'(y'_i)$ and $css(v) = css'(v')$.*

Proof. Assume, without loss of generality, that the machine M contains only one channel.

For the first part we construct v' from v as follows: replace each transition in v by its corresponding pair of transitions from Eq. (1). Replace each "message loss" transition in v by a pair of message losses, one for the message and one for the $\#$ that follows that message.

The proof for the second part is slightly more technical, and is proved by induction on the length of v'.

The hypothesis is true for the base case when $|v'| = 0$. It suffices to choose $v = \varepsilon$.

For the induction case, assume that the hypothesis is true for $|v'| \leqslant l$, $l \geqslant 0$, and consider the case where $|v'| = l + 1$. We will express $v' = v'_1 v'_2$, and will consider four cases depending upon v'_2. ∎

Case of $v'_2 = L$. In this case we have $v' = v'_1 L$ and $\langle p, x'\rangle \xrightarrow{v'_1}_{M'} \langle q, z'_1 a z'_2\rangle \xrightarrow{L}_{M'} \langle q, z'_1 z'_2\rangle$. By induction hypothesis, there exists a v_1 such that $\langle p, x\rangle \xrightarrow{v_1}_M \langle q, f'(z'_1 a z'_2)\rangle$ and $css(v_1) = css'(v'_1)$. Since $f'(z'_1 z'_2) \preceq f'(z'_1 a z'_2)$ we can extend the execution sequence of M to $\langle p, x\rangle \xrightarrow{v_1}_M \langle q, f'(z'_1 a z'_2)\rangle \xrightarrow{v_2}_M \langle q, f'(z'_1 z'_2)\rangle$ where $v_2 \in L^*$. We have $css(v_1 v_2) = css'(v'_1 v'_2)$ for $css(v_2) = css'(v'_2) = \varepsilon$.

Case of $v'_2 = D$. Is similar to the previous case but we use the fact that $f'(z'_1 a z'_2) = f'(z'_1 a a z'_2)$ and we choose $v_2 = \varepsilon$.

Case of $v'_2 = (p', c!a, r_t)\ v'_3(r_t, c!\#, q)$ and $v'_3 \in (L \cup D)^*$. There are three subcases.

The first subcase is when the message a sent at the beginning of v'_2 is dropped by a "message loss" transition in v'_3. In this case, we can write $v'_3 = v'_4 L v'_5$ where L is responsible for deleting the a. Since the execution sequence $v'_2 = (p', c!a, r_t)\ v'_4 L v'_5 (r_t, c!\#, q)$ can be reordered to $(p', c!a, r_t)\ v'_4 v'_5 (r_t, c!\#, q)\ L$ without affecting the final state reached, this subcase can be handled by the first case we have considered.

The second subcase is when the message a is duplicated, and this subcase is similar to the previous one.

The final subcase is when the message a is neither lost nor duplicated by any action in v'_3. In this case the execution sequence can be written as

$$\langle p, x'\rangle \xrightarrow{v'_1}_{M'} \langle p', z'_1\rangle \xrightarrow{(p', c!a, r_t)}_{M'} \langle r_t, z'_1 a\rangle$$
$$\xrightarrow{v'_3}_{M'} \langle r_t, z'_2 a\rangle \xrightarrow{(r_t, c!\#, q)}_{M'} \langle q, z'_2 a\#\rangle.$$

But we can reorder this execution sequence to

$$\langle p, x'\rangle \xrightarrow{v'_1}_{M'} \langle p', z'_1\rangle \xrightarrow{v'_3}_{M'} \langle p', z'_2\rangle$$
$$\xrightarrow{(p', c!a, r_t)}_{M'} \langle r_t, z'_2 a\rangle \xrightarrow{(r_t, c!\#, q)}_{M'} \langle q, z'_2 a\#\rangle.$$

That is, move all the duplication and lossiness actions before sending the a. Since the actions of v'_3 do not affect the a, their actions on the buffer contents z'_1 results in the buffer being z'_2.

By the induction hypothesis, since $|v'_1 v'_3| < |v'|$ there is an execution sequence v_1 of M such that $\langle p, x\rangle \xrightarrow{v_1}_M \langle p', f'(z'_2)\rangle$ such that $css(v_1) = css'(v'_1 v'_3)$.

There are now tow subcases depending upon whether $f'(z'_2) = zb$, $b \neq a$, or $f'(z'_2) = za$. In the former case we can extend the execution sequence v_1 of M as

$$\langle p, x\rangle \xrightarrow{v_1}_M \langle p', f'(z'_2)\rangle \xrightarrow{(p', c!a, q)}_M \langle q, f'(z'_2)\, a\rangle = \langle q, f'(z'_2 a \#)\rangle$$

and in the latter case we have the following execution sequence in M:

$$\langle p, x\rangle \xrightarrow{v_1}_M \langle p', f'(z'_2)\rangle \xrightarrow{(p', c!a, q)}_M \langle q, f'(z'_2)a\rangle \xrightarrow{L}_M \langle q, f'(z'_2)\rangle = \langle q, f'(z'_2 a \#)\rangle.$$

Since $css((p', c!a, q)) = css((p', c!a, q)\, L) = css'((p', c!a, r_t)\, v'_3(r_t, c!\#, q))$ the conclusion holds.

Case of $v'_2 = (p', c?a, r_t)\, v'_3(r_t, c?\#, q)$ *and* $v'_3 \in (L \cup D)^*$: is similar to the previous case, and hence omitted.

COROLLARY 4. *There exists no algorithm which when given a machine capable of both lossiness and duplication errors can compute its reachability set.*

Proof. Let us use f' for n-tuple of words; define $f'(\langle x_1, ..., x_n\rangle) = \langle f'(x_1), ..., f'(x_n)\rangle$. By the fact that $f'(a_1 \# a_2 \# \cdots a_m \#) = a_1 a_2 \cdots a_m$ and from Lemma 8 we have $L_M(s) = f'(L_{M'}(s))$ for each $s \in S$. Then the conclusion follows from Lemma 7, Theorem 7 and recognizability of $L_{M'}(s)$ (Theorem 4 and Lemma 6). ▌

COROLLARY 5. *There is no algorithm to model-check a machine capable of lossiness and duplication against a specification in CTL*.*

Proof. Assume that there is a procedure to test given a formula in *CTL** and an arbitrary machine capable of lossiness and duplication whether the formula holds for that machine.

Let s be a state of machine M; now consider the property $\phi(s)$ for the machine M' (constructed in Lemma 8):

$$\phi(s) = E(GF(@s) \wedge \neg FG(@s)).$$

The property $\phi(s)$ says that there is an execution sequence in which the control resides at state s infinitely often, but it is not the case that there is a infinite tail where the control is always at state s. Clearly from Lemma 8, an infinite execution sequence v' satisfying $\phi(s)$ exists in M' iff there is an infinite execution sequence v in M in which s is visited infinitely often (as $css' = v') = css(v)$, in this case). Consequently, if model-checking is decidable for the class of machines with both lossiness and duplication errors then the RSP problem is solvable for machines capable of lossiness. ▌

To summarize, we have:

THEOREM 16. *For the class of machines that are capable of lossiness and duplication errors, we have that the reachability set is not computable and the model-checking problem is not decidable.*

7. CONCLUSION

We summarize the results known to date, on unreliable channels, in the Table 1. Apart from being almost complete, it does provide a comparison of the expressive power of the three kinds of errors. Clearly duplication has no effect on the expressive power. Lossiness on the other hand makes the communicating machines less powerful. What is surprising is that insertion makes the communicating machines even less powerful, as a description of the set of all reachable states can be calculated for machines with insertion errors but not for machines with lossiness errors. Pachl proved in [15] that if the reachability set is recognizable than the reachability problem is decidable. What we have shown is that this result does not scale up to other problems, and perhaps surprisingly, even though the reachability set might

TABLE 1

A Survey of Decidability Results

	Lossy	Csp	Dup	Insert	Lossy & Dup	Lossy & Insert	Dup & Insert
Reachability	D	?	U	D	D	D	D
Boundedness	U(?)	U(?)	U	D	D	D	D
Deadlock	D	D	U	D	D	D	D
Finite Termination	D	D	U	D	D	D	D
Model Checking against *CTL**	U	U	U	D	U	D	D
Is Reachability set recognizable?	Yes	?	No	Yes	Yes	Yes	Yes
Computating finite automaton for reachability set	U	U_c	n/a	D	U	D	D

Note. D—Decidable, U—Undecidable, U(?)—conjectured to be undecidable, n/a—not applicable, Csp—completely specified protocols, U_c—undecidable provided channel language is recognizable.

be recognizable a machine-independent description need not be computable.

The contributions of this paper are new results for verification of communication machines whose channels have duplication error, insertion errors, or a combination of duplication, insertion and lossiness errors. These results are significant in that assumptions about the possibility of such errors (which are closer to reality) make the verification problems easier. Finally, we have also presented the decidability results for all three kinds of errors (duplication, insertion, and lossiness) and their combination with in a single framework.

Received June 28, 1994; final manuscript received June 28, 1995

REFERENCES

1. Abdulla, P., and Jonsson, B. (1993). Verifying programs with unreliable channels, *in* "Proceedings of Logic in Computer Science," Eighth Annual Symposium on Logic in Computer Science, Montreal, June 1993, Computer Science Press.
2. Abdulla, P., and Jonsson, B. (1994). Undecidability of verifying programs with unreliable channels, *in* "Proceedings of ICALP," LNCS 820.
3. Bartlett, K. A., Scantlebury, R. A., and Wilkinson, P. T. (1969). A note on reliable full-duplex transmission over half-duplex lines, *Comm. ACM* **12**(5), 260–265.
4. Berstel, J. (1979). "Transductions and Context-Free Languages," Teubner, Stuttgart.
5. Bochmann, Gregor (1978). Finite state description of communication protocols, *Comput. Networks* **2**, 362–372.
6. Brand, Daniel, and Zafiropulo, Pitro (1983). On communicating finite-state machines, *J. Assoc. Comput. Mach.* **30**(2), 323–342.
7. CCITT Recommendation Z.100 (1988). "Specification and Description Language SDL," Blue Book Vols. X.1–X.5, ITU General Secretariat, Geneva.
8. Diaz, M., Ansart, J. P., Azema, P., and Chari, V. (1989). "The Formal Description Technique Estelle," North-Holland, Amsterdam.
9. Emerson, E. A., and Halpern, J. Y. (1986). "Sometimes" and "not never" revisited: On branching time versus linear time temporal logic, *J. Assoc. Comput. Mach.* **33**(1), 151–178.
10. Finkel, Alain (1994). Decidability of the termination problem for completely specified protocols, *Distrib. Comput.* **7**, 129–135.
11. van Glabbeek, R. J., and Vaandrager, F. W. (1993). Modular specification of process algebras, *Theoret. Comput. Sci.* **113**, 293–348.
12. Gouda, M. G., Gurari, E. M., Lai, T.-H., and Rosier, L. E. (1987). On deadlock detection in systems of communicating finite state machines, *Comput. Artif. Intell.* **6**(3), 209–228.
13. Higman, G. (1952). Ordering by divisibility in abstract algebras, *Proc London Math. Soc.* **2**, 326–336.
14. Lothaire, M. (1983). "Combinatorics on Words," Addison–Wesley, Reading, MA.
15. Pachl, J. K. (1987). Protocol description and analysis based on a state transition model with channel expressions, *in* "Proceedings of Protocol Specification, Testing and Verification, VII, May," North-Holland.
16. Peng, W., and Purushothaman, S. (1991). Data flow analysis of communicating finite state machines, *ACM Trans. Programming Languages Systems* **13**(3), 399–442.
17. Tarski, A. (1955). A lattice theoretic fixpoint theorem and its application, *Pacific J. Math.* **5**, 285–305.