Duplication, Insertion and Lossiness errors in Unreliable Communication Channels

Gérard Cécé Alain Finkel
LIFAC, ENS de Cachan,
61 av. du Pdt. Wilson
94235 CACHAN Cedex, France

S. Purushothaman Iyer
Dept of Computer Science,
North Carolina State University,
Raleigh, NC 27695-8206, USA

Abstract

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 regarding verification of FIFO channels that can lose messages have been considered by Finkel [10], and by Abdulla and Johnson [1, 2]. 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 is a model of computation that forms the backbone of ISO standard protocol specification languages Estelle and SDL. While an 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.

Keywords: Finite State Machines; Communication Channels; Duplication, Insertion and Lossiness errors; Verification problems; model-checking against CTL*; Decidability.

1 Introduction

Finite state machines which communicate over unbounded channels have been used as an abstract model of computation for reasoning about communication protocols [4, 11] and form the backbone of ISO protocol specification languages Estelle [8] and SDL [7]. Ever since the publication of the Alternating bit protocol [3] (the first ever 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 [5].

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 entire state space of such machines can be expressed as regular languages. Furthermore, we show how to compute a description of this regular language. 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. A complementary view, on errors, to what has already been proposed in the literature. In particular we consider insertion errors, duplication errors and combinations of the three (insertion, duplication and lossiness) errors.
2. A comparison of the relative expressive power of these errors. Our findings are that insertion errors decrease 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

SIGSOFT '94- 12/94 New Orleans LA USA
© 1994 ACM 0-89791-691-3/94/0012..$3.50

recall the necessary mathematical definitions. In Section 3 we offer an overview of past work. We present our results for arbitrary insertion errors in Section 4, our results for duplication in Section 5 and the various combinations in Section 6. In the conclusion we offer a comparison of the relative expressive power of these errors.

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 Figure 1. We could as well consider the single machine acting on two channels c_1 and c_2, as shown in Figure 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 turing machines [5]. Formally, we have

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

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

is the transition relation. ■

Notations 1 We will use $x \cdot y$ to emphasize the concatenation of strings x and y. Let $\Sigma_C = \bigcup_{c \in C} \Sigma_c$. ■

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 $M = (S, C, \Sigma_C, \delta)$ the set of reachable states of the machine M is the least set $R(M) \subseteq S \times \Sigma_1^* \times \ldots \times \Sigma_n^*$ defined inductively by the following rules:

- (initial state axiom) $\langle s_0, \varepsilon, \ldots, \varepsilon \rangle \in R(M)$ – the initial state.
- If $u = \langle s, x_1, \ldots, x_n \rangle \in R(M)$ then
  - (output rule) If $(s, c_i!a, s') \in \delta$ then

    $$u' = \langle s', x_1, \ldots, x_{i-1}, x_i \cdot a, x_{i+1}, \ldots, x_n \rangle \in R(M).$$

    This defines the semantics of an output action. Furthermore, u' is said to be a successor of u.
  - (input rule) If $(s, c_i?a, s') \in \delta$ and $x_i = a \cdot x_i'$ then

    $$u'' = \langle s', x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n \rangle \in R(M).$$

    This defines the semantics of an input action. Furthermore, u'' is said to be a successor of u.

[1] We use the words processes and machines synonymously

■

In the future we call such machines which do not model errors as normal machines.

Notations 2 Let $\sum_1^n = \Sigma_1^* \times \ldots \Sigma_n^*$, the set of all n-tuple of words over the channel alphabets. ■

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 3 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

$$L(s) = \{\langle x_1, \ldots, x_n \rangle \in \sum_1^n | \langle s, x_1, \ldots, x_n \rangle \in R(M)\}$$

■

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

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

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

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

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

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

■

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

Theorem 1 (Brand et al [5]) *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 5 Let $x, y \in \Sigma^*$. $x \preceq y$ (i.e., x is a subword of y) provided $x = a_1 \ldots a_n$ and $y = y_0 a_1 y_1 \ldots a_n y_n$ where $y_i \in \Sigma^*$ and $a_i \in \Sigma$. ■

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

Theorem 2 (Higman [12, 13]) *If a set of words W consists of mutually incomparable elements according to $\preceq$ then W is finite.*

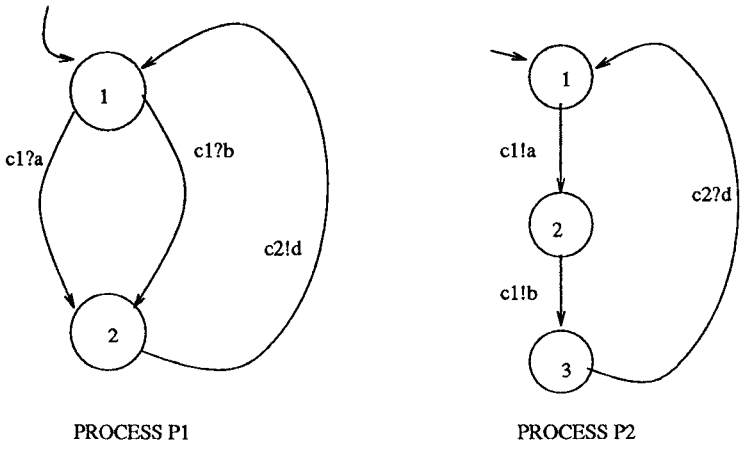

Figure 1: Two communicating machines

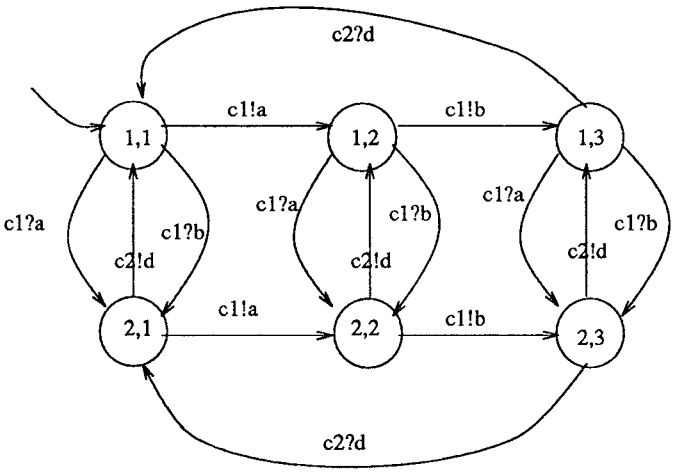

Figure 2: Alternative view: One machine acting on two buffers

We can, without loss of generality, also use $\preceq$ for n-tuples of words; define $\langle x_1, \ldots, x_n \rangle \preceq \langle y_1, \ldots, y_n \rangle$ provided for every i, such that $1 \leq i \leq n$, we have $x_i \preceq y_i$. Given a set $W \subseteq \sum_1^n$, we will say that W is upward closed provided for each $w \in W$ every element w' such that $w \preceq w'$ is also in W. Formally,

Definition 6 Let $W \subseteq \sum_1^n$. Define $closure(W) = \{w' \in \sum_1^n | w \preceq w' \wedge w \in W\}$. Consequently, a set of n-tuples of words W is said to be upward closed provided

$$W = closure(W)$$

■

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 of minimal elements of any upward closed set is necessarily finite. Given a minimal element $w = a_1, \ldots, a_n$, it is easy to see that if w is in a upward closed set W then $\Sigma^* a_1 \Sigma^* \ldots \Sigma^* a_n \Sigma^*$ is also included in the set W. Consequently, we can look upon these minimal elements as a representation of that set. For a upward closed set W we will write $min(W)$ for the minimal elements of W. As there is a finite set of generators for every upward closed, we have:

Lemma 1 *Every upward closed set is regular.* ■

Consider two related upward closed sets, then there is a relation between the minimal elements of those two sets:

Lemma 2 *Let $A \subseteq B$ and let both sets be upward closed. We then have*

$$\forall a \in min(A)\, \forall b \in min(B) \text{ either } b \preceq a \text{ or } (a \not\preceq b \text{ and } b \not\preceq a)$$

■

Notations 3 $A \subseteq B$ provided A is a subset of B, and $A \subset B$ provided A is properly contained in B. ■

A consequence of the above lemma is that given an upward closed set A there is no infinite sequence of successively larger sets (that are all bigger than A). Formally, we have

Lemma 3 *Let A be an upward closed set. There is no infinite sequence of upward closed sets $\{A_i\}_{i \geq 0}$ such that*

$$A \subset A_0 \subset A_1 \ldots$$

Proof: Given an upward closed set A assume that there is an infinite sequence, $\{A_i\}_{i \geq 0}$, of successively larger sets. Now consider the sequence $\wp = \{w_i\}_{i \geq 0}$ formed as follows: (a) initially populate $\wp$ with elements from $min(A)$, (b) successively consider each of the $A_i, i \geq 0$, and add to $\wp$ those elements of $min(A_i)$ which are not related to any element added to $\wp$ thus far. By construction, this sequence contains mutually incomparable elements. By Higman's Lemma, this sequence is necessarily finite. Without loss of generality, assume that $\wp$ terminates when elements from $min(A_\ell)$ are being added to $\wp$. By our previous lemma all elements in $min(A_i), i > \ell$, have to be less than some element in $\wp$. Since $b \preceq a$ implies that the length of b is less than or equal to the length of a, there is a finite $j \geq \ell$ such that all $A_i, i \geq j$, are the same, contradicting our assumption.

Consequently, given an upward closed set A the sequence

$$A \subset A_0 \subset A_1 \subset \ldots$$

of upward closed sets is finite. ■

3 Previous 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 where as 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. We first recall their definitions:

Definition 7 A completely specified protocol is a machine $M = (S, C, \Sigma_C, s_0, \delta)$ whose reachability set is the least set $R_{csp}(M) \subseteq S \times \sum_1^n$ defined by the following rules:

- The initial state axiom, output rule and input rule are as for normal machine.
- $\forall i : 1 \leq i \leq n$ and $\forall a \in \Sigma_{c_i}$ we have
  if $u = \langle s, x_1, \ldots, x_{i-1}, a x_i, x_{i+1}, \ldots, x_n \rangle \in R_{csp}(M)$
  then $u''' = \langle s, x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n \rangle \in R_{csp}(M)$.
  Again u''' is said to be a successor of u.

■

An execution path of a machine M is a sequence

$$\{\langle p_i, x_{i1}, \ldots, x_{in} \rangle\}_{i \geq 0}$$

such that (a) $p_0 = s_0$, $x_{0j} = \varepsilon$, for all $j : 1 \leq j \leq n$ and (b) every element of the sequence is a successor of the previous element in that sequence.

Definition 8 A machine M is said to finitely terminate provided every execution path starting from $\{s_0, \varepsilon, \ldots, \varepsilon\rangle\}$ is finite. ■

Finkel, in [10], showed the following:

Theorem 3 *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 $R_L(M) \subseteq S \times \sum_1^n$ defined by the following inductive rules:

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

■

The notion of successor states carries over from the definition of completely specified protocol.

Abdulla and Jonsson show the following in [1, 2]:

Theorem 4 *The following are true for machines capable of lossiness errors:*

- *The reachability problem and the deadlock problems are decidable.*
- *The model-checking problem is undecidable.*
- *The channel language is regular.*

The following facts were not reported earlier by Finkel, or Abdulla and Jonsson, though these follow from their theorems:

Theorem 5 (Cécé et al [6]) *Given a machine M*

- *$R_L(M)$ is finite iff $R_{csp}(M)$ is finite.*
- *$R_L(M)$ has a deadlock state iff $R_{csp}(M)$ has a deadlock state.*

Unlike the previous theorem which is a direct consequence of the definitions from [10, 1, 2], the following new undecidability result is due to a reduction from a problem shown to be undecidable in [2]: the Recurrent Path Problem (RPP). This problem decides the existence of an infinite execution path which visits infinitely often a given state s.

Theorem 6 (Cécé et al [6]) *Given a machine M*

> *There exists no procedure to compute a finite state machine (or a regular expression) representation of $R_L(M)$.*

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 arbitrary insertion errors has the reachability set $R_I(M) \subseteq S \times \sum_1^n$ 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.
- $\forall 1 \leq i \leq n$ and $\forall a \in \Sigma_i$ we have
  if $\langle s, x_1, \ldots, x_{i-1}, x_i' x_i'', x_{i+1}, \ldots, x_n \rangle \in R_I(M)$
  then $\langle s, x_1, \ldots, x_{i-1}, x_i' a x_i'', x_{i+1}, \ldots, x_n \rangle \in R_I(M)$

■

A machine that is capable of arbitrary insertion errors can insert any message, at any location of the queue. While this is very close to our mental notion of arbitrary errors, it is technically much more easier to deal with a slightly different, but equivalent, machine model. In this slightly altered model, called *insertion machine*, arbitrary unspecified messages can *only be inserted at the end of the queue*. The advantage is that in the insertion machine there is a bound on the number of successor to any states, but in the case of arbitrary insertions there is no such bound (as the number of successors of a state depends upon the length of the words in that state).

Definition 11 A machine $M = (S, C, \Sigma_C, \delta)$ is an insertion machine provided

$$\forall s \in S, \forall c \in C, \forall a \in \Sigma_c . (s, c!a, s) \in \delta$$

Given a machine $M = (S, C, \Sigma_C, \delta)$ its insertion completion is the machine $I(M) = (S, C, \Sigma_C, \delta \cup \{(s, c!a, s) | s \in S, c \in C, a \in \Sigma_c\})$. ■

But the two models are equivalent in that their reachability sets are the same. Formally, we have:

Lemma 4 *Let M be a machine, $R_I(M)$ be the reachability set of M when it is capable of arbitrary insertion errors and let $R(I(M))$ be the reachability set of the insertion completion of M. We have*

$$R_I(M) = R(I(M))$$

Proof: The inclusions in both direction can be proven by an induction on the length of the justification that a particular state is reachable. ■

4.1 Computation of $R(M)$ for a Insertion Machine

We will now show how to compute the set $R(M)$ when M is an insertion machine. The computation will yield a machine independent characterization (i.e., a regular expression) in terms of minimal elements of an upward closed set. The calculation itself involves setting up a set of equations and solving them. We need the following operations to state the equations:

Definition 12 Let $W \subseteq \sum_1^n$. Define

$$(c_i!a)(W) = \left\{ \begin{array}{l} \langle w_1, \ldots, w_{i-1}, w_i \cdot a, w_{i+1}, \ldots w_n \rangle | \\ \langle w_1, \ldots, w_{i-1}, w_i, w_{i+1}, \ldots w_n \rangle \in W \end{array} \right\}$$

Similarly define

$$(c_i?a)(W) = \left\{ \begin{array}{l} \langle w_1, \ldots, w_{i-1}, w_i, w_{i+1}, \ldots w_n \rangle | \\ \langle w_1, \ldots, w_{i-1}, a \cdot w_i, w_{i+1}, \ldots w_n \rangle \in W \end{array} \right\}$$

■

Given an insertion machine $M = (S, C, \Sigma_C, s_0, \delta)$, in this section we will show how to compute a description of the reachability set, $R(M)$, of M. By the construction of the insertion machines, the channel languages are upward closed.

Lemma 5 *For every state $s \in S$ of an insertion machine M, we have $L(s)$ is upward closed. Consequently, for all $s \in S$, $L(s)$ is a regular set.* ■

To compute the channel language we can state equations between the various channel languages and solve them. The flow equations, which are similar to what was used in [15], are as follows:

$$\forall s \neq s_0 : \quad L(s) = \bigcup_{(s', c!a, s) \in \delta} (c!a)(L(s')) \cup \bigcup_{(s', c?a, s) \in \delta} (c?a)(L(s')) \qquad (1)$$

$$L(s_0) = \langle \epsilon, \ldots, \epsilon \rangle \cup \bigcup_{(s', c!a, s_0) \in \delta} (c!a)(L(s')) \cup \bigcup_{(s', c?a, s_0) \in \delta} (c?a)(L(s')) \qquad (2)$$

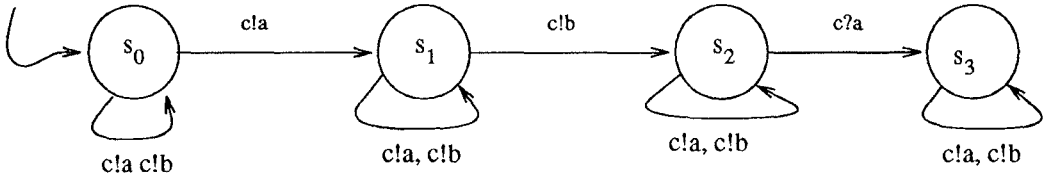

Figure 3: An insertion machine

Given that the eventual solution for each of the variables $L(s), s \in S$, is an upward closed set we can restate the equations as:

$$\forall s \neq s_0 : \; L(s) = closure\left(\begin{array}{l} \bigcup_{(s',c!a,s)\in\delta}(c!a)(L(s'))\cup \\ \bigcup_{(s',c?a,s)\in\delta}(c?a)(L(s')) \end{array} \right) \tag{3}$$

$$\begin{aligned} L(s_0) &= closure\left(\begin{array}{l} \langle\varepsilon,\ldots,\varepsilon\rangle\cup \\ \bigcup_{(s',c!a,s_0)\in\delta}(c!a)(L(s'))\cup \\ \bigcup_{(s',c?a,s_0)\in\delta}(c?a)(L(s')) \end{array} \right) \\ &= \textstyle\sum_1^n \end{aligned} \tag{4}$$

Given equation (4) it is tempting to think that the channel language for every state would be $\sum_1^n$. But this is not so, as the channel language for states s_1, s_2 and s_3 of the machine in Figure 3 does not include ε.

The first thing to notice in these equations is that the domain of the variables is the powerset of $\sum_1^n$, which is a complete lattice. Furthermore the operators involved in these equations, $c!a$, $c?a$, $\cup$ and *closure* are continuous operators over the domain of the powerset of $\sum_1^n$. Consequently, these equations have an unique least solution. Due to Tarski's Fixed-Point theorem [16] the computation of this least solution can be expressed as the limit (viz., union) of a sequence of solutions as follows:

$$\begin{array}{l} \forall s \neq s_0 : \; L(s)_0 = \emptyset \\ \forall i \geq 0 : \; L(s_0)_i = \sum_1^n \\ \forall s \neq s_0, \forall i \geq 0 \text{ we have} \\ \quad L(s)_{i+1} = closure\left(\begin{array}{l} \bigcup_{(s',c!a,s)\in\delta}(c!a)(L(s')_i)\cup \\ \bigcup_{(s',c?a,s)\in\delta}(c?a)(L(s')_i) \end{array} \right) \end{array} \tag{5}$$

An immediate consequence of the equations (5) is:

Lemma 6 *For the equations (5) of an insertion machine the following hold:*

1. $\forall s \in S$ *and* $\forall i \geq 0$ *we have* $L(s)_i \subseteq L(s)_{i+1}$.
2. $\exists$ *a finite* ℓ *such that* $\forall s \in S, i \geq \ell$. $L(s)_i = L(s)_\ell$.

Consequently, the solution to equations (5) can be computed in a finite number of iterations.

Proof: The first part follows by the monotonicity of the operators involved in the equations, and the second part follows from Lemma 3. ■

Since we can compute the channel language for every state of the machine, all of our problems of interest (deadlock, reachability, unboundedness etc) can be immediately solved. To check whether a state $\langle s, x_1, \ldots, x_n\rangle$ is reachable we need to check whether $\langle x_1, \ldots, x_n\rangle$ is in the channel language of s, i.e. $L(s)$. Note that no state can be deadlock state because there are always transitions out of every state. Furthermore, every machine is obviously unbounded as an unbounded number of messages can be inserted into the buffer. Consequently,

Theorem 7 *We have the following for machines capable of insertion errors:*

- *The channel language is computable.*
- *The reachability problem is decidable.*
- *The deadlock and boundedness problems are trivially solvable.*

Proof: The proof follows from Lemmas 4, 5 and 6. ■

5 Duplication errors

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

Definition 13 A machine $M = (S, C, \Sigma_C, s_0, \delta)$ is capable of duplication errors when its reachability set $R_D(M)$ is defined as the least set satisfying the following rules:

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

■

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 (as many as there are edges in the machine), and always sending a

after sending a normal message. More pictorially, the transformations look as follows:

$$p \xrightarrow{c!a} q \text{ is transformed to } p \xrightarrow{c!a} p' \xrightarrow{c!\#} q$$

and

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

where p' is a new state not in S.

We therefore need to 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 as 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. For every non-duplicate language L define

$$L_D = \{a_1^+ \dots a_n^+ | a_1 \dots a_n \in L\}$$

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

$f(\varepsilon) = \varepsilon$
$f(a) = a, \forall a \in \Sigma$
$f(x_1 a a x_2) = f(x_1 a x_2), \forall x_1, x_2 \in \Sigma^*, a \in \Sigma$
$f(x_1 a b x_2) = f(x_1 a) f(b x_2), \forall x_1, x_2 \in \Sigma^*, a, b \in \Sigma$ *and* $a \neq b$

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

Lemma 7

$$x \in L \quad iff \quad x \in L_D \text{ and } f(x) = x$$

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 know to be recursively enumerable, we have:

Theorem 8 *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, 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:

$$if \langle p, x_1, \dots, x_n \rangle \in R_L(M) \text{ then } \langle p, x_1', \dots, x_n' \rangle \in R_L(M)$$
$$\forall x_i' \preceq x_i, 1 \leq i \leq n$$

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

Lemma 8 *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:*

- $L(s) = \sum_1^n$ *iff there is a sequence of transitions from s_0 to s.*
- $L(s) = \emptyset$ *or* $L(s) = \sum_1^n$.

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 9 *For a machine that has both lossiness and insertion errors the reachability, 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 from 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 10 *We have the following for machines capable of insertion errors and duplication errors:*

- *The channel language is computable.*
- *The reachability problem is decidable, and the deadlock 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, most of the work for lossy machines [1, 2] is applicable to machines that have both lossiness and duplication errors.

Definition 14 Given a machine $M = (S, C, \Sigma_C, s_0, \delta)$ define the duplication completion $D(M)$ as follows:

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

■

	Lossy	Dup	Insert	Lossy&Dup	Lossy&Insert	Dup&Insert
Reachability	D	U	D	D	D	D
Boundedness	U(?)	U	D	D	D	D
Deadlock	D	U	D	D	D	D
Model Checking against CTL*	U	U	D	U	D	D
Is Reachability set regular?	Yes	No	Yes	Yes	Yes	Yes
Computating Finite Automaton for Reachability set	U	n/a	D	U	D	D

Legend: D – Decidable, U– Undecidable, U(?) – conjectured to be undecidable, n/a – not applicable.

Figure 4: A survey of decidability results

As in the case of insertion machine, the following holds:

Lemma 9 *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 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. In fact, all of the decidability results of machines with lossiness errors also holds for machines that can have duplication and lossiness errors. Furthermore, boundedness is decidable for lossiness and duplication, where as it is still an open problem for lossy systems (clearly, one can duplicate a letter in the buffer to produce buffers of unbounded size). Undecidability results also carry over though their proofs are not straightforward; the proofs appear in the complete paper. Summarizing, we have:

Theorem 11 (Cécé et al [6]) *The reachability problem and the boundedness problem for a machine that has both duplication and lossiness errors is solvable. Furthermore, the channel language is not computable and model checking against CTL* is undecidable.*

7 Conclusion

We summarize the results known to date, on unreliable channels, in the Table 4. 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 [14] that if the reachability set is regular 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 be regular 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.

References

[1] P. Abdulla and B. Jonsson. Verifying Programs with Unreliable Channels. In Proc of *Logic in Computer Science*, 1993.

[2] P. Abdulla and B. Jonsson. Undecidability of verifying programs with unreliable channels. To appear in Proc of *ICALP*, 1994.

[3] K. A. Bartlett, R. A. Scantlebury and P. T. Wilkinson. A note on reliable full-duplex transmission over half-duplex lines. *CACM*, 12(5):260–265, 1969.

[4] Gregor Bochmann. Finite State Description of Communication Protocols. *Computer Networks*, 2:362–372, 1978.

[5] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *JACM*, 30(2):323–342, 1983.

[6] G. Cécé, A. Finkel and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. LIFAC Tech Report 94-02. ENS de Cachan, France, May 1994.

[7] CCITT Recommendation Z.100: *Specification and Description Language SDL*, Blue Book Vol X.1-X.5, 1988, ITU General Secretariat, Geneva.

[8] M. Diaz, J. P. Ansart, P. Azema, and V. Chari. *The Formal Description Technique Estelle.* North Holland, 1989.

[9] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: on branching time versus linear time temporal logic. *JACM*, 33(1):151–178, 1986.

[10] Alain Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7:129–135, 1994.

[11] 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.

[12] G. Higman. Ordering by divisibility in abstract algebras. In Proc. of *London Math Society,* 2:326–336, 1952.

[13] M. Lothaire. *Combinatorics on Words.* Addison-Wesley, 1983.

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

[15] W. Peng and S. Purushothaman. Data flow analysis of communicating finite state machines. *ACM Transactions on Programming Languages and Systems,* 13(3):399–442, July 1991.

[16] A. Tarski. A Lattice Theoretic Fixpoint Theorem and its Application. *Pacific Journal of Mathematics,* 5:285–305, 1955.