Simulating Perfect Channels with Probabilistic Lossy Channels [2]

Parosh Abdulla [a] Christel Baier [b] S. Purushothaman Iyer [c,1] Bengt Jonsson [a]

[a] *Dept of Computer Systems, Uppsala University, SE-751 05 Uppsala, Sweden*

[b] *Dept of Computer Science, University of Bonn, 53117 Bonn, Germany*

[c] *Dept of Computer Science, North Carolina State University, Raleigh, NC 27695, USA*

Abstract

We consider the problem of deciding whether an infinite-state system (expressed as a Markov chain) satisfies a correctness property with probability 1. This problem is, of course, undecidable for general infinite-state systems. We focus our attention on the model of *probabilistic lossy channel systems* consisting of finite-state processes that communicate over unbounded lossy FIFO channels. Abdulla and Jonsson have shown that safety properties are decidable while progress properties are undecidable for non-probabilistic lossy channel systems. Under assumptions of "sufficiently high" probability of loss, Baier and Engelen have shown how to check whether a property holds of probabilistic lossy channel system with probability 1.

In this paper we consider a model of probabilistic lossy channel systems, where messages can be lost only during send transitions. In contrast to the model of Baier and Engelen, once a message is successfully sent to channel, it can only be removed through a transition which receives the message. We show that checking whether safety properties hold with probability 1 is undecidable for this model. Our proof depends upon simulating a perfect channel, with a high degree of confidence, using lossy channels.

[1] Supported in part by ARO under grant DAAG55-98-1-03093 and DAAD-19-01-1-0683, by NSF under 0098037, by European Commission FET project ADVANCE, contract No IST-1999-29082, and by STINT.

[2] A version of this paper appeared in the proceedings of CONCUR 2000.

1 Introduction

Finite state machines which communicate over unbounded FIFO channels have been used as an abstract model of computation for reasoning about communication protocols [6], and they form the backbone of ISO protocol specification languages Estelle and SDL. However, the model is Turing-powerful which makes most verification problems of interest undecidable. Ever since the publication of the Alternating bit protocol it has been customary to assume, while modeling a protocol, that the communication channels between processes are free of errors. Possible errors in the communication channels are treated separately, or are completely ignored. However, over the past few years there have been attempts to rectify this situation to allow modeling of imperfections in the communication medium. Abdulla and Jonsson, in [2], considered a model where messages could be lost from the queue, while waiting in a queue to be delivered. They showed that the *reachability* problem (and consequently, the problem of checking for safety properties) is decidable. However, in [1], they also showed that checking for progress properties is undecidable – a consequence of fairness arguments necessary to prove progress properties. Randomization is an oft-used technique to provide tractable, approximate solutions to intractable problems or to supplant fairness arguments, necessary for showing progress properties, by probabilistic arguments [11]. Given that we are dealing with imperfections in the communication medium it is then natural to consider models of communicating processes where the probability of message loss is taken into account. Thus, by considering a probabilistic model we would be making our model more closer to reality. In [8], Iyer and Narasimha considered whether a probabilistic lossy channel systems satisfies an LTL formula with a probability greater than p, with in a tolerance ϵ. In [4] Baier and Engelen considered the following problem (which is the topic of this paper):

> given a probabilistic lossy channel system $\mathcal{L}$ and a linear-time temporal logic formula ϕ, does the set of sequences of $\mathcal{L}$ that satisfy ϕ have probability 1.

A partial answer to this question was given by Baier and Engelen who showed that it is decidable if the probability of message loss is sufficiently high. Here "sufficiently high" was taken to mean roughly "at least $\frac{1}{2}$". Under this restriction, it can be shown that (with probability 1) the number of messages in the channel cannot grow unboundedly, and that the general model checking problem can be reduced to checking reachability (which is decidable by [2]).

In this paper we consider a problem related to that left open by Baier and Engelen. We show that the problem of checking safety properties with probability 1 is undecidable, when we consider a slightly different semantical model. In contrast to the model of [4], where any message inside a channel my chosen and removed (lost) from the channel, we only allow loss of messages during send transitions. More precisely, each time a message is sent to channel, it is either lost or appended to the end of the channel. Once the message is added to the channel, it can only be removed through a transition which receives the message.

Our proof is a construction which shows that a lossy channel system can (with probability 1) simulate a finite state machine which operates on *perfect* FIFO channels. The main idea is to let each message transmission of the perfect channel system be simulated by a number of retransmissions of the message in the lossy channel system. If the scheme for retransmissions is chosen appropriately, the simulation will be faithful in a certain well-defined sense – a fact which allows us to establish the undecidability result. To our knowledge this is the first undecidability proof in reasoning about probabilistic computational models and, thus, the construction should be of independent interest.

The rest of the paper is organized as follows. In the next two sections we present the necessary definitions for probabilistic lossy channel systems and the probabilistic reachability problem. In Section 4 we provide an overview of our undecidability proof, which we follow in Section 5, Section 6, and Section 7 with the necessary constructions. These constructions are used in Section 8 to show the correctness theorem. Finally, we give some conclusions in Section 9.

2 Communicating Finite-state Machines

In this section we introduce communicating finite-state machines (CFSMs) and some of their properties.

For a set M we use M^* to denote the set of finite strings of elements in M. For $x, y \in M^*$ we let $x \bullet y$ denote the concatenation of x and y. The empty string is denoted by ε. For sets C and M, a *string vector from C to M* is a function in $C \mapsto M^*$. For a string vector w from C to M we use $w[c := x]$ for the string vector w' such that $w'(c) = x$, and $w'(d) = w(d)$, for $d \neq c$. The string vector which maps all elements in C to the empty string is denoted ε.

Definition 1 *A* Communicating Finite-State Machine (CFSM) $\mathcal{C}$ *is a tuple* $\langle S, C, M, \delta \rangle$*, where*

- S *is a finite set of* control states*,*
- C *is a finite set of* channels*,*
- M *is a finite set of* messages*,*
- δ *is a finite set of* transitions*, each of which is a triple of the form* $\langle s_1, op, s_2 \rangle$*, where* s_1 *and* s_2 *are control states, and* op *is either a label of form* $c!m$*,* $c?m$*,*

or empty, where $c \in C$ *and* $m \in M$*.*

Given this finite description of a system, we can now talk about the *global state* γ of $\mathcal{C}$ as a pair $\langle s, w \rangle$, where $s \in S$ and w is a string vector from C to M. The *initial global state* γ_{init} of $\mathcal{C}$ is the pair $\langle s_{init}, \varepsilon \rangle$. The progression of the system from one global state to another can now be formalized as a transition relation $\longrightarrow$ which is a set of triples of the form $\langle \gamma_1, t, \gamma_2 \rangle$, where γ_1 and γ_2 are global states and $t \in \delta$. The relation $\gamma_1 = \langle s_1, w_1 \rangle \xrightarrow{t} \gamma_2 = \langle s_2, w_2 \rangle$ holds iff one of the following conditions is satisfied.

(1) $t = \langle s_1, c!m, s_2 \rangle$ and $w_2 = w_1[c := w_1(c) \bullet m]$ (a *send* operation).
(2) $t = \langle s_1, c?m, s_2 \rangle$ and $w_1 = w_2[c := m \bullet w_2(c)]$ (a *receive* operation).
(3) $t = \langle s_1, \mathit{empty}, s_2 \rangle$ and $w_2 = w_1$ (an *empty* operation).

For a global state γ, we define the set $en(\gamma)$ of *enabled* transitions at γ such that $t \in en(\gamma)$ iff $\gamma \xrightarrow{t} \gamma'$ for some γ'. We let $\longrightarrow = \cup_{t \in \delta} \xrightarrow{t}$ and let $\xrightarrow{*}$ denote the reflexive transitive closure of $\longrightarrow$. For global states γ_1 and γ_2, we say that γ_2 is *reachable* from γ_1 if $\gamma_1 \xrightarrow{*} \gamma_2$. For a global state γ and a control state s, we say that γ is *reachable* from s if there is w such that $\langle s, w \rangle$ is reachable form γ.

A *computation* π is a sequence (either finite or infinite) of states and transitions: $\gamma_1 \; t_1 \gamma_2 \; t_2 \; \gamma_3 \; \cdots \; t_{n-1} \gamma_n \ldots$ such that $\gamma_i \xrightarrow{t_i} \gamma_{i+1}$ for each $i : 1 \leq i$. We will use $\pi(i)$ to denote the i^{th} state γ_i visited in π. If a computation π is finite we say that π *leads* from γ_1 to γ_n [3].

We define the control state reachability problem for CFSMs (Reach-CFSM) as follows

Instance A CFSM $\mathcal{C}$ and a control state s_F in $\mathcal{C}$.
Question Is s_F reachable from γ_{init}?

The following result is well-known (e.g. [6]).

Theorem 2 *Reach-CFSM is undecidable.*

The idea behind the proof is to use one of the channels to simulate the tape of a Turing machine.

Remark To simplify the undecidability proof, we consider a restricted class of CFSM. Undecidability of Reach-CFSM holds also for the restricted class. Therefore, the restricted class does not imply loss of generality. We assume that a CFSM satisfies the following four conditions

[3] Note that γ' is reachable from γ iff there is a computation leading from γ to γ'.

- We assume that a CFSM has only one channel. The undecidability proof of Reach-CFSM needs only one channel which is used to simulate the tape of a Turing machine.
- It will never occur that two copies of the same message m are placed beside each other inside a channel. In other words, we will never reach a configuration where the content of the channel is of the form $M^* \bullet m \bullet m \bullet M^*$. Given a CFSM $\mathcal{C}$, we can derive a new CFSM $\mathcal{C}'$ satisfying this property by adding a new sumbol # to the alphabet, and splitting each send transition $t = \langle s_1, c!m, s_2 \rangle$ into two transitions $t_1 = \langle s_1, c!m, s_t \rangle$ and $t_2 = \langle s_t, c!\#, s_2 \rangle$, where s_t is a new control state. Furthermore, each receive transition $t = \langle s_1, c?m, s_2 \rangle$ is split into two transitions $t_1 = \langle s_1, c?m, s_t \rangle$ and $t_2 = \langle s_t, c?\#, s_2 \rangle$.
- After each receive operation, there will be at least one message left in the channel. In other words, no receive operation will make the channel empty. We can derive a CFSM with this property by first sending two symbols (say $\#_1$ and $\#_2$) to the channel, and then running the original CFSM. Each control state is provided with a loop consisting of four transitions, performing the following operations: receive $\#_1$, send $\#_1$, receive $\#_2$, send $\#_2$. These transitions are used to move back the symbols $\#_1$ and $\#_2$ from the head to the end of the channel to enable receive transitions in the original CFSM.
- No control state is the source of only receive transitions, i.e., there is a send or an empty transition out of every control state. This restriction is achieved by adding self-loops that are empty transitions to those control states that are source of only receive transitions.

In the sequel we assume that all CFSMs satisfy the above four properties.

3 Probabilistic Lossy Channel systems

In this section we consider probabilistic lossy channel systems (PLCSs).

Definition 3 *A* lossy channel system (LCS) $\mathcal{L}$ *is of the same form as a CFSM with multiple channels. However, the transition relation $\longrightarrow$ is extended such that $\langle s_1, w \rangle \xrightarrow{t} \langle s_2, w \rangle$ if $t = \langle s_1, c!m, s_2 \rangle$. In other words, we allow a message sent to the channel to be lost without ever being appended to the contents of the channel.*

It should be noted that an LCS need not satisfy the restrictions described in Section 2. Another point to note is that the semantics of loss used here is slightly different from the one used in [2,8,4], where a message can be lost at any time from the queue. In contrast, in the current paper a message can only be lost when it is being placed in the queue. However, the set of reachable states under both semantics is the same [2].

Definition 4 *A* probabilistic Lossy Channel system (PLCS) *is a tuple*

$$\langle S, C, M, \delta, \mathsf{W}, \lambda \rangle$$

where $\langle S, C, M, \delta \rangle$ *is an LCS,* λ *is real number in the interval* $[0, 1]$ *representing the probability of losing messages, and* W *is a* weight function *which assigns to each transition* $t \in \delta$ *a positive real number* $\mathsf{W}(t)$.

Given that a PLCS is also an LCS we will lift the transition relation $\longrightarrow$, from LCS, to obtain the transition relation for PLCS by normalizing the weights assigned to transitions. More precisely, for a global state γ and a transition t we define $\mathsf{W}'(\gamma)(t)$ to be equal to $\mathsf{W}(t)$ if $t \in en(\gamma)$ and to be equal to 0 otherwise. We define $P(\gamma_1 \xrightarrow{t} \gamma_2)$, where $\gamma_1 = \langle s_1, w_1 \rangle$ and $\gamma_2 = \langle s_2, w_2 \rangle$, to be equal to

- $(1 - \lambda) \cdot (\mathsf{W}'(\gamma_1)(t) / \sum_{t' \in \delta} \mathsf{W}'(\gamma_1)(t'))$, if t is of the form $\langle s_1, c!m, s_2 \rangle$ and $w_1 \neq w_2$. This corresponds to performing a non-lossy send operation.
- $\lambda \cdot (\mathsf{W}'(\gamma_1)(t) / \sum_{t' \in \delta} \mathsf{W}'(\gamma_1)(t'))$, if t is of the form $\langle s_1, c!m, s_2 \rangle$ and $w_1 = w_2$. This corresponds to a lossy send operation.
- $(\mathsf{W}'(\gamma_1)(t) / \sum_{t' \in \delta} \mathsf{W}'(\gamma_1)(t'))$, if t is of the form $\langle s_1, c?m, s_2 \rangle$ or of the form $\langle s_1, empty, s_2 \rangle$.

Notice that $P(\gamma_1 \xrightarrow{t} \gamma_2)$ is well-defined only if $\sum_{t' \in \delta} \mathsf{W}'(\gamma_1)(t') \neq 0$. This can be guaranteed if from each control state there is at least one transition with a send or an empty operation. The PLCS we shall describe in the next sections satisfies this property.

For a finite computation $\pi = \gamma_1\ t_1\ \gamma_2\ t_2\ \gamma_3\ \cdots\ t_{n-1}\ \gamma_n$, we define

$$\mathcal{B}_\pi = \{\pi' | \pi \text{ is a prefix of } \pi'\}$$

as a basic cylinder set with probability $P(\mathcal{B}_\pi) = \prod_{0<i<n} P(\gamma_i \xrightarrow{t_i} \gamma_{i+1})$. We will assume the standard measure space based on the Borel field generated from these basic cylinder sets [10] by taking closure under denumerable unions, denumerable intersection and complementation. For global states γ_1 and γ_2, we define $P(\gamma_1, \gamma_2) = P\left(\bigcup_{\pi \text{ leads from } \gamma_1 \text{ to } \gamma_2} \mathcal{B}_\pi\right)$. For a global state γ and a control state s, we define $P(\gamma, s) = P(\bigcup_{\pi \text{ leads from } \gamma \text{ to some } \langle s,w \rangle} \mathcal{B}_\pi)$.

The reachability problem for PLCS (Reach-PLCS) is defined as follows:

Instance A PLCS $\mathcal{L}$ and control state s_F in $\mathcal{L}$.
Question Is $P(\gamma_{init}, s_F) = 1$?

Note that checking safety properties can be reduced to checking reachability of a bad control state [2]. Furthermore, checking for liveness properties amounts to checking for repeated reachability of a control state [4]. Consequently, our attention to the probabilistic reachability problem is appropriate.

4 Overview of our Undecidability Proof

In this section, we give an overview of the undecidability proof for Reach-PLCS. The undecidability result is achieved through a reduction from Reach-CFSM.

Suppose that we are given an instance of Reach-CFSM, i.e., a CFSM $\mathcal{C}$ and a control state s_F. Recall that we assume that $\mathcal{C}$ has a single channel, which we will call c_M. We shall construct an equivalent instance of Reach-PLCS, i.e., a PLCS $\mathcal{L}$ which "simulates" $\mathcal{C}$ such that s_F is reached with probability 1 if and only if s_F is reachable in $\mathcal{C}$. The PLCS $\mathcal{L}$ has a lossy channel[4] c_M which simulates the channel in $\mathcal{L}$. In addition, $\mathcal{L}$ has two other channels which will be described below. The main idea of the construction is to implement two protocols which allow $\mathcal{L}$ to simulate the computations of $\mathcal{C}$:

- One of the protocols generates multiple copies of transmitted messages, and guarantees there is a positive probability of simulating a perfect computation. Consequently, if s_F is not reachable in $\mathcal{C}$ then there is a positive probability that it will not be reachable in $\mathcal{L}$.
- The second protocol restarts the system from time to time and guarantees that if s_F is reachable in $\mathcal{C}$ then it will be reachable in $\mathcal{L}$ with probability one.

Generating multiple copies Due to the risk of losing messages, each send operation $c_M!m$ of $\mathcal{C}$ is simulated by a sequence of retransmissions of the message m in $\mathcal{L}$. This means that each receive operation must be simulated by a "receive loop" where all copies of a message are received.

Retransmitting a message several times decreases the probability that the message is lost in the channel (i.e., that all copies of the message are lost), but the probability is still nonzero. We will therefore construct a scheme in which the number of retransmissions of a message is increased as the execution proceeds. The rate of the increase is sufficiently large to guarantee a positive probability that no message is "completely lost" during the simulation. In

[4] Notice that we use c_M both for the channel in $\mathcal{C}$ and that in $\mathcal{L}$.

other words, for each message, at least some copies[5] will be maintained in the channel. The ability to construct a perfect simulation of a CFSM by a PLCS (with a probability greater than zero) is a central concept in the proof. If s_F is not reachable in $\mathcal{C}$ then, with a positive probability, s_F will not be reachable in $\mathcal{L}$.

Restarting The simulation process consists of a number of *rounds*. Each round consists of "restarting" $\mathcal{C}$. We will devise a mechanism whereby the simulation is "restarted" periodically. The period between two restarts determines the number of steps of $\mathcal{C}$ which are simulated during the particular round. The rounds should be longer and longer so that eventually they are sufficiently long to simulate an entire computation from γ_{init} to s_F in $\mathcal{C}$ (in case such a computation exists). Notice that this implies that $\mathcal{L}$ should never deadlock, since this would mean that the restarting procedure cannot be continued.

This protocol ensures that if s_F is reachable in $\mathcal{C}$ then s_F is will be reachable in $\mathcal{L}$ with a probability one: $\mathcal{L}$ is restarted infinitely often (we will run infinitely many rounds). When the rounds get sufficiently long, the state s_F will be reached in the current round with a probability which is bounded from below.

To carry out the two protocols just outlined the PLCS $\mathcal{L}$ uses, in addition to c_M, two lossy channels called the *retransmission counter* c_T and the *resetting counter* c_S.

The counter c_T The number of retransmissions (copies) of each message will be controlled by the counter c_T. This counter is implemented by a (lossy) channel with two messages, 0 and 1 (say). When the simulation of a send operation is about to start, the contents of c_T will be either of the form 0^k or of the form 1^k. The integer k denotes a counter value k. A transition t corresponding to sending a message m in $\mathcal{C}$ is replaced in $\mathcal{L}$ by a sequence of operations. We describe this sequence of operations when the counter is of the form 0^k. We receive all the 0s in c_T. We replace each copy of 0 by a number of copies of 1 in c_T and m in c_M. More precisely, for each 0 received from c_T we send r copies of 1 to c_T and send r copies of m to c_M (the choice of r is described in Section 7). This means that while sending the copies of m, the content of the counter will be of the form $0^{k_1}1^{k_2}$, where k_1 is the number of times the copying procedure has still to be repeated[6]. This operation will

[5] For technical reasons related to our construction, we need at least two copies of each message to be maintained. See the description of the recieve operation in Section 5.
[6] $value(c_T = 0^{k_1}1^{k_2}) = k_1$ denoting the current value of c_T

continue until a 1 is received, indicating that all 0s have been received. At this point the content of c_T will be of the form 1^ℓ. In total, the above procedure sends $k \cdot r$ copies of 1 to c_T and $k \cdot r$ copies of m to c_M.

The sequence of operations is similar when we start from a configuration where the counter is of the form 1^k. The difference is that we now receive 1s from c_T and send 0s.

Observe that the contents of c_T is always either of the form $0^{k_1}1^{k_2}$ in which case we say that the counter is in *mode* 0, or of the form $1^{k_1}0^{k_2}$ in which case we say that the counter is in *mode* 1.

The counter c_S The mechanism for restarting the simulation must be devised so that the system $\mathcal{L}$ is restarted infinitely often, with increasing periods between the restarts. This is controlled by the counter c_S. The counter c_S should satisfy two properties

- Its value should be increasing at successive restarting points. In fact, its value should increase much more quickly than the value of c_T (in a sense made more precise in Section 5).
- Its value should be decreasing between two restarting points.

The two conditions are not contradictory (although they might seem to be). During a simulation, we continuously decrease the current value of the counter. Just before the next round starts, the counter is assigned a value which is exponential in the current value of c_T.

When a new round is initiated, the contents of c_S is either of the form 0^k or of the form 1^k. The value k of counter c_S represents the number of steps for which the system $\mathcal{C}$ is simulated, before the next restart is performed. Suppose that the content of c_S is of the form 0^k. When simulating a step of $\mathcal{C}$ the current value of c_S is decreased by receiving a 0. Also, from time to time we send a 1 to c_S. This means that, during the simulation procedure, the content of c_S will be of the form $0^{k_1}1^{k_2}$, where k_1 is the current value of the counter (the number of steps of $\mathcal{C}$ that still have to be simulated before the current round is terminated and the next round is initiated). In implementing c_S, we shall guarantee that its value will decrease even if the rest of the system has deadlocked. Although $\mathcal{C}$ will never deadlock by assumption, $\mathcal{L}$ may deadlock. This can happen, for instance, when c_M is empty and $\mathcal{L}$ tries to receive a message.

In a similar manner to the counter c_T, the counter c_S is either of the form $0^{k_1}1^{k_2}$ in which case we say that the counter is in *mode* 0, or of the form $1^{k_1}0^{k_2}$ in which case we say that the counter is in *mode* 1.

Channel Clean-Up Each time a new round is about to be started we need the channel c_M to be empty, since we are simulating a computation of $\mathcal{C}$ which starts from an empty channel. To achieve that we use a *clean-up* procedure which removes all messages left inside c_M from the previous round. To implement the clean-up operation, we use two new symbols $\$_0$ and $\$_1$ not in M as end markers. The clean-up operation is either in *mode* 0 in which case the content of c_M is of the form $\$_0^* M^* \$_1^*$, or in mode 1 in which case the content of c_M is of the form $\$_1^* M^* \$_0^*$. In mode 0 (mode 1) the clean-up operation is performed by receiving messages from c_M until we receive a $\$_1$ (a $\$_0$). This means that there are no members of M left in c_M after the clean-up operation is done.

5 Implementation of Operations

In this section we show in greater detail how to implement $\mathcal{L}$ such that its behaviour satisfies the properties described in Section 4. The control states in $\mathcal{L}$ are the following:

- Each control state in $\mathcal{C}$ has a copy in $\mathcal{L}$.
- There are two special states: *exit* from which the restarting procedure is started, and *clean* from which the clean-up procedure is started.
- There are a number of "intermediate" states which do not correspond to any state in $\mathcal{C}$, and which are used in the simulations of the operations.

Furthermore, there are three flags f_T, f_S, and f_C. These are Boolean variables representing the mode of counter c_T, the mode of counter c_S, and the mode of the clean-up operation. Obviously, the three flags can be encoded in the control part of $\mathcal{L}$ by making eight copies of all control states and letting each copy represent one combination of the values of the three flags. However, for simplicity of presentation we model the flags as Boolean variables.

In the text below we describe the operations assuming that the flags f_T, f_S, and f_C are all equal to 0. The behaviour during other modes can be derived by substituting appropriate values for the three flags.

Clean-up This operation (Figure 1) resets the content of c_M. Assuming mode 0, the content of c_M is of the form $\$_0^* M^* \$_1^*$. We start from *exit* and send messages of form $\$_1$ to c_M, while simultaneously receiving messages from c_M until we receive a $\$_1$. In such a case we know that the content of c_M is a string in $\$_1^*$. The mode is now changed to 1. The reason why we need two symbols (rather than only one) is the fact that some end markers may be left inside the channel from the previous clean-up operation. By alternating the use of end

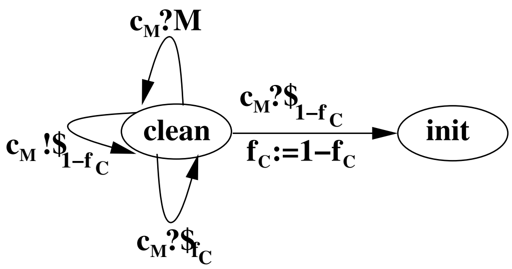

Fig. 1. The clean-up operation. Note: $c?M$ should be read as a set of receive commands, each receiving a member of M.

markers we can distinguish between the current end marker and the previous one.

Initial states Define Γ_{init} to be the set of states of form $\langle s_{init}, w\rangle$ such that $w(c_M)$ is a string in either $\$_0^*$ or $\$_1^*$, i.e., c_M does not contain any messages from M. The members of Γ_{init} are the configurations from which the different rounds are started.

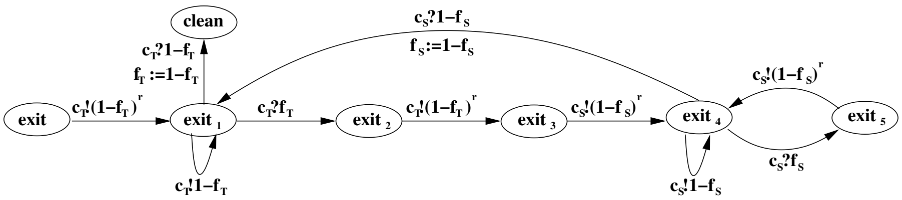

Fig. 2. The restarting operation

Restarting The aim of the restarting procedure (Figure 2) is to increase the value of the counter c_T, and also to make the value of c_S exponentially larger than the value of c_T. First, we send r copies of 1 to c_T. Then, we iterate two nested loops: an outer loop (from $exit_1$ through $exit_2$ and $exit_3$ back to $exit_1$), and an inner loop (from $exit_4$ through $exit_5$ back to $exit_4$). Each time the outer loop is iterated we increase the value of c_T by r while we multiply (through the inner loop) the value of c_S by r.

Send: A send transition of the form $t = (s, c_M!m, s')$ in $\mathcal{C}$ is simulated in $\mathcal{L}$ by the sequence of transitions depicted in Figure 3. First, r copies of 1 and

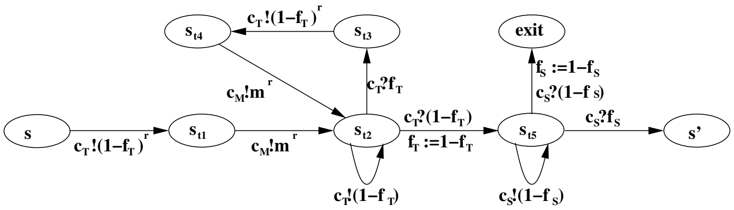

Fig. 3. Translation of send transition $\langle s, c_M!m, s'\rangle$. Note: $c_T!(1-f_T)^r$ should be read as r commands, each sending a copy of $1-f_T$ to c_T. For instance, if f_T is zero then the above operation corresponds to sending r copies of 1 to c_T.

r copies of m are generated. Thereafter the loop from from s_{t2} via s_{t3} and s_{t4} generates r times as many copies of 1 as the value of c_T (i.e., number of copies of 0 in c_T) together with an equal number of copies of m. The loop is repeated until we receive a 1 from c_T, indicating that we have consumed all the 0s in c_T, whereupon we change mode of c_T and move to state s_{t5}. Observe that c_T will now contain a string of 1s. At state s_{t5}, we receive a message from c_S. If no 0s are left in c_S, i.e., if there is a 1 at the head of c_S, we change the mode of c_S and move to state $exit$, thus initiating the restart procedure. Otherwise, we receive a 0 from c_S and proceed to state s', where we continue

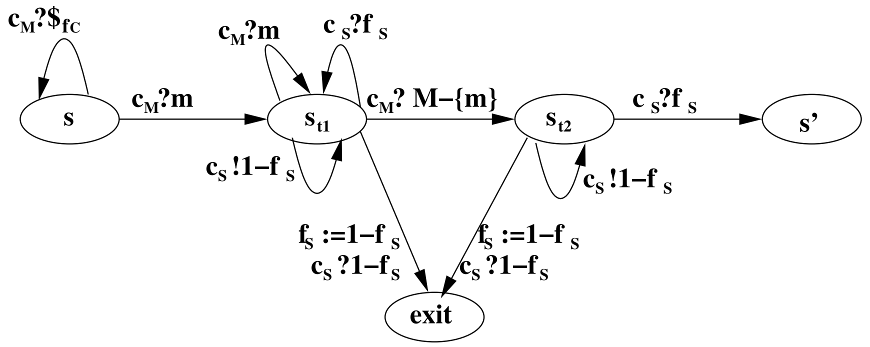

Fig. 4. Translation of receive transition $\langle s, c_M?m, s'\rangle$. Note: We use similar notation here as in Figure 3. By $c_M?M - \{m\}$ we mean a set of commands, each receiving a message in the set $M - \{m\}$.

with the simulation. The aim of the self-loops from states s_{t2} and s_{t5} is to prevent deadlocks when trying to receive from c_T and c_S, respectively.

Receive: A receive transition of form $t = \langle s, c_M?m, s'\rangle$ is simulated as in Figure 4. First the self-loop at s receives copies of $\$_0$ that may have been left from the last clean-up operation. Thereafter, we receive m from c_M and move to s_{t1}. At s_{t1} there are a number of self-loops. One of these receives copies of m from c_M, until receiving a message which is different from m and moving to s_{t2}. By the assumption that $\mathcal{C}$ will never put two successive copies of m into c_M, receiving a message different from m indicates that all copies of m have been consumed, provided that the simulation has been faithful since the last cleanup operation. From s_{t2}, we decrease the value of c_S, move to s', and continue with the simulation. The two transitions from s_{t1} that receive from c_S prevent deadlock in case no messages different from m is left in c_M. Note that these transitions are enabled even when m is present. The aim of the self-loops at states s_{t1} and s_{t2} that transmit 1s to c_S is to prevent deadlocks when trying to receive from c_M and c_S.

The self-loop labeled with $c_s?f_S$ from s_{t_1} is enabled even in the case where there are available copies of m in c_M. This may cause too many restarts since all copies of c_S may be recieved before all copies of m are consumed. In order to avoid this, we make the value of c_S grow more quickly than that of c_T (in a sense which we shall make more precise in Section 7).

Notice that the transition from s_{t_1} to s_{t_2} removes one copy of the next message to be received. Therefore, we require the copying protocol to preserve at least two copies of each message (rather than only one).

Empty transition: The translation for an empty transitions is similar to the translation of a send transition. The only difference is that no messages will be sent to c_M. In other words, we merge the states s_{t4} and s_{t2} and remove the transition between these two states.

It should be noted that the translation of an empty transition in $\mathcal{C}$ can not be an empty transition in $\mathcal{L}$. This is due to the fact that $\mathcal{C}$ can get caught in a self-loop of empty transitions (say) at a state s, if the only other possible transition from s attempts to receive a message type that is not in front of the queue c_M. While $\mathcal{C}$ could be stuck, its simulation $\mathcal{L}$ should recover to start a new simulation, of $\mathcal{C}$, by driving the counter c_S to zero.

6 Properties of Operations

In this section, we prove properties of the operations which we later use in the correctness proof.

We assume all transitions in $\mathcal{L}$ are of equal weight. This is not an essential assumption. In fact our construction works provided that all weights are positive.

We first state two lemmas describing a number of simple properties about the control flow of the operations. The lemmas follow by inspecting the operations in Figures 1, 2, 3, and 4.

Lemma 5 (Invariants) *The following properties are invariants of any computation of $\mathcal{L}$.*

- *The contents of channel c_T is always in $(f_T)^*(1 - f_T)^*$, where f_T is the current mode of c_T.*
- *The contents of channel c_S is always in $(f_S)^*(1 - f_S)^*$, where f_S is the current mode of c_S.*
- *If f_C is the current mode of the cleanup operation then the contents of channel c_M is in $\$^*_{f_C} M^*$, except in control state clean where it is in $\$^*_{f_C} M^* \$^*_{1-f_C}$. where*

Lemma 6 (Progress) *The following progress properties hold for the simulating operations.*

- *Whenever the clean-up procedure starts in control state clean, with probability 1 it leads to control state init with the contents of c_M in $\$^*_{f_C}$.*
- *Whenever the restart procedure is started in control state exit, with probability 1 it leads to control state clean.*
- *Whenever a send, receive, or empty operation is started in a control state s of $\mathcal{C}$, with probability 1 it leads either to control state exit or to its final control state s' of $\mathcal{C}$.*

A sequence π^i_j of transitions is called a *send segment* if it is an execution of the translation of a send transition (from control state s to either *exit* or s' as in Figure 3). We define *receive segment*, *empty segment*, *restart segment*, and *cleanup segment* analogously. We say that a send segment π^i_j is *faithful* if at least two copies of the retransmitted message m are not lost in c_M during π^i_j.

A *round* is a sequence of transitions which is a concatenation of form $\pi_1 \cdots \pi_n \pi_{rs} \pi_{cu}$ where

- π_1 starts in a state in Γ_{init}
- each $\pi_1, \ldots, \pi_n$ is send, receive, or empty segment,
- π_{rs} is a restart segment, and π_{cu} is a cleanup segment,
- π_{cu} ends in a state in Γ_{init}

We use Lemma 5 and Lemma 6 to determine the structure of computations of $\mathcal{L}$.

Lemma 7 *with probability 1, a computation π of $\mathcal{L}$ starting from Γ_{init} is a concatenation*

$$\pi \quad = \quad \pi^1_1 \pi^1_2 \pi^1_2 \cdots \pi^1_{n_1} \pi^1_{rs} \pi^1_{cu} \quad \pi^2_1 \pi^2_2 \cdots \pi^2_{n_2} \pi^2_{rs} \pi^2_{cu} \quad \cdots \quad \pi^i_1 \cdots \pi^i_{n_i} \pi^i_{rs} \pi^i_{cu} \quad \cdots$$

of an infinite sequence of finite rounds.

PROOF. The lemma follows from the following arguments:

- After any segment of form π^i_j with $j < n_i$ or of form π^i_{cu}, the PLCS $\mathcal{L}$ is in a control state s of $\mathcal{C}$. By the assumption that no control state s in $\mathcal{C}$ is the source of only receive transitions, some send, receive, or empty segment will be initiated. By Lemma 6, each such segment terminates with probability 1, either in a control state s' of $\mathcal{C}$ or in the state *exit*. If it leads to *exit*, the next segment is a restart segment which by Lemma 6, with probability 1 leads to *clean*, followed by a cleanup segment which with probability 1 leads back to a state in Γ_{init}.
- Each restart segment terminates with a specific value of c_S, and each send, receive, and empty segment (except possibly $\pi^i_{n_i}$) reduces the current value of c_S by at least one. Once the current value of c_S becomes zero, the control state *exit* is entered. Hence there can be only a finite number of send, receive, and empty segments between two consecutive restart segments. By Lemma 6, with probability 1 the last cleanup operation leads to a state in Γ_{init}. As a consequence of Lemma 6, with probability 1 each round terminates in a state in Γ_{init}. $\square$

It is easy to verify from the construction of $\mathcal{L}$ that if all send segments are faithful, then each control state of $\mathcal{C}$ which is visited by $\mathcal{L}$ during π is indeed reachable in $\mathcal{C}$.

7 Properties of Counters

In this section we analyze the behavior of the counters c_T and c_S.

Recall that λ is the probability of losing a message, and that r decides the rate by which c_T increases (and hence the number of copies of each message sent to c_M during the simulation of a send operation). We choose r such that $r \cdot (1 - \lambda) > 2$.

We recall and prove some properties of stochastic processes.

Branching Processes A *branching process X with parameters r and p* represents a sequence of *generations* as follows. In the 0th generation, there is one individual. For each $n \geq 0$, each individual in the nth generation generates r individuals in the $(n+1)$-st generation, each of which survives with probability $p : 0 < p < 1$ (and dies with probability $1-p$). Thus, each individual in the n-th generation generates a number of surviving individuals in the

$(n+1)$-st generation, according to a binomial distribution with parameters r and p. For a branching process X, we let X_n be the random variable which gives the number of individuals in the n-th generation. Notice that X_0 is one with probability one, and that X_1 has a binomial distribution.

The following lemma is shown in [9] and states that if the expected number of survivors from the offsprings of an individual is strictly greater than one then there is a positive probability that each generation will have at least a single survivor.

Lemma 8 *Let X be a branching process with parameters r and p. If $r \cdot p > 1$ then there is a positive probability that $\forall i.\ X_i > 0$.*

From Lemma 8 we get:

Corollary 9 *Let X be a branching process with parameters r and p. If $r \cdot p > 1$ then, for each $K > 0$, there is a positive probability that $\forall i.\ (i \geq K \implies X_i \geq K)$.*

PROOF. There is a positive probability that $X_K \geq K$. Consider the members of the K-th generation. Each of them can be viewed as the originator of a new branching process. This means that the members of the next generations (after K) can be viewed as a stochastic variable which is the sum of at least K stochastic variables each of which is a branching process with parameters r and p. By Lemma 8 each of these branching processes will survive with a positive probability ρ. This means that, with probability at least ρ^K, we have $X_i \geq K$ for all $i \geq K$. $\square$

The following lemma states that if the expected number of survivors from the offsprings of an individual is strictly greater than two then there is a positive probability that sizes of successive generations will increase exponentially. The proof of the lemma (which makes use of Chebyshev's inequality) is given in the appendix.

Lemma 10 *Let X be a branching process with parameters r and p. If $r \cdot p > 2$ then there is a positive ρ such that for all i, the probability that $X_i \geq 2^i$ is greater than ρ.*

Negative Binomial Distribution Suppose that an infinite sequence of independent experiments is conducted. The outcome of each experiment is either *success*, with probability p, or *failure*, with probability $q = 1 - p$. Let X be a random variable denoting the number of failures before obtaining y successes, with $y > 0$. Then, X is said to have a *negative binomial distribution with parameters y and p.*

In the appendix we show the following.

Lemma 11 *Consider $p : 0 < p < 1$. There is a natural number α and a positive rational ρ such that the following holds: Let $y > 0$ and let X be a stochastic variable which has a negative binomial distribution with parameters y and p. Then, the probability of $X \leq \alpha \cdot y$ is greater than ρ.*

Counter $\mathbf{c_T}$ We consider the behaviour of counter c_T. We observe that the value of c_T is changed only in send, empty, and restart segments. Also, the mode of c_T is changed by either by the transition $exit_1$ to $clean$ in Figure 2, or the transition from s_{t_2} to s_{t_5} in Figure 3 (or the corresponding transition in an empty segment). Consider counter c_T just before a mode-changing (i.e., before a transition of one of the above three forms is executed). Assuming mode f_T, the contents of c_T is then of the form $(1 - f_T)^k$. We observe that k is the new value which will be assigned to c_T. Let $k_T^0\ k_T^1\ k_T^2\ \cdots$ be the number of messages (0s or 1s) appended to c_T between the successive mode switches. We observe that we obtain k_T^{i+1} from k_T^i by sending $r \cdot k_T^i$ new messages to c_T. More precisely, in Figure 3, the message removed in the transition from s_{t_2} to s_{t_3} is replaced by r new messages in the transition from s_{t_3} to s_{t_4}. Also, the message removed in the transition from s_{t_2} to s_{t_5} (or from $exit_1$ to $clean$) is replaced by r new messages in the transition from s to s_{t_1} (or from $exit$ to $exit_1$). The latter addition of messages is performed next time we enter a send, empty, or restart segment [7].

Observe that there is a positive probability that $k_T^0, , k_T^1 \geq 2$ since $r \cdot (1 - \lambda) > 2$ and consequently $r \geq 2$. Furthermore, it is evident from the above discussion that we can view $k_T^0\ k_T^1\ k_T^2\ \cdots$ as a branching process. From this and Corollary 9 we get the following.

Lemma 12 *Let $k_T^0\ k_T^1\ k_T^2\ \cdots$ be the successive values of c_T between mode switches of c_T.*

(a) There is a positive probability that $\forall i.\ k_T^i \geq 2$.
(b) For any positive K, with probability 1 there are infinitely many i such that $k_T^i \geq K$.

PROOF. As described above, we can view $k_T^0\ k_T^1\ k_T^2\ \cdots$ as a branching process. Consequently, claim (a) follows immediately from Corollary 9 and the fact that there is a positive probability that $k_T^0, k_T^1 > 0$.

To prove claim (b), we observe that, if $k_T^i = 0$ then a new branching process is initiated. This is achieved by the transition $exit$ to $exit_1$ in Figure 2, or the transition from s to s_{t_2} in Figure 3 (or the corresponding transition in an empty

[7] Also, additional messages are added in the self-loops from $exit_1$ and s_{t_2}.

segment). Since each of these branching processes survives with a probability which bounded from below, one of them will survive with probability one. Claim (b) follows immediately. □

Since the number of copies of sent messages is equal to the number of copies sent to c_T (Figure 2 and Figure 3), we can state a similar lemma about the number of copies of messages transmitted to c_M.

Lemma 13 *There is a positive probability that all send segments are faithful.*

The following lemma defines an upper bound on the growth of c_T.

Lemma 14 *For each n there are positive ρ and α, such that the following holds. Consider a computation π of the form*

$$\pi_{rs}\pi_{cu}\pi_1\pi_2\cdots\pi_n$$

where $\pi_1,\ldots,\pi_n$ are send, empty, or receive segments. Suppose that k_T is the value of c_T when π is started . Then, with probability at least ρ the value of c_T never exceeds $k_T\cdot(r+\alpha)^{n+1}+r$ during π.

PROOF. In π_{rs} we first send r new massages to c_T (the transition from *exit* to $exit_1$). Then, each time we remove a copy of f_T in π we send r copies of $(1-f_T)$ to c_T (see the discussion before Lemma 12). Also, we send a number of copies of $(1-f_T)$ to c_T in the self-loops from $exit_1$ in Figure 2 and s_{t_2} in Figure 3. We estimate the number of copies of $(1-f_T)$ that can be sent in these self-loops as follows. We observe that the number of messages sent in each self-loop is determined by a sequence of independent experiments that are performed, which with probability 0.5 result [8] in a self-loop transmitting $(1-f_T)$ and with probability 0.5 result in receiving f_T. Let k'_T be the value of c_T when a (send, empty, or restart) segment is started. The number of copies of $(1-f_T)$ transmitted in the corresponding self-loop is then the number of self-loop outcomes that occur before k'_T reception outcomes. By Lemma 11, there are positive ρ_1 and α, which are independent of k'_T, such that, with probability at least ρ_1, this number is at most $\alpha\cdot k'_T$. Define $\rho=\rho_1^{n+1}$. □

Counter $\mathbf{c_S}$ We consider counter c_S. We show that it grows exponentially more quickly than c_T.

[8] We get probability 0.5 since all transitions are assumed to have equal weights. See Section 6.

Lemma 15 *There is a positive ρ, such that whenever a restart segment is started in control state exit with the value of c_T being k_T, then with probability at least ρ the restart segment leads to state clean where the value of c_S is at least $2^{k_T} - 1$.*

PROOF. During the restart procedure, each time we remove a copy of f_T, we eventually move to $exit_4$ and perform a number of iterations of the inner loop. In a similar manner to the analysis of the behaviour of c_T, we can view the behaviour of c_S during these iterations as a branching process. We observe that the number of iterations is equal to k_T and that the last time the transition from $exit_4$ to $exit_1$ is performed the value of c_S is decreased by one. By Lemma 10 it follows that there is a positive probability ρ that the value of c_S is at least $2^{k_T} - 1$ when the restarting procedure is concluded (and we enter control state *clean*). $\square$

We finally need to state a lemma about how a receive segment affects counter c_S. Potentially, a receive segment can decrease the current value of c_S very much before consuming all copies of m. This may cause c_S to become 0 and drive $\mathcal{L}$ into the exit state, starting the restart procedure before a desired number of steps of $\mathcal{C}$ have been simulated. We therefore need to prove a bound on the speed at which c_S is decreased, in relation to the number of copies of m in c_M.

Lemma 16 *Suppose that, upon starting a receive segment sequence at control state s, the number of copies of the message m in c_M is k_M and the value of c_S is k_S, with $k_S > k_M$. Also, suppose that the sequence of copies of m is followed by a least one member of $M - \{m\}$ available in c_M. Then with probability 0.5, the segment enters state s' with the value of c_S being at least $k_S - k_M$.*

PROOF. A receive operation will either lead to s' because it successfully simulates a receive transition, or lead to *exit*. The simulation can enter *exit* because either (i) there is no member of $M - \{m\}$ which follows the copies of m (and therefore the transition from s_{t_1} to s_{t_2} is not enabled); or (ii) the current value of c_S will be reduced to 0. In case a member of $M - \{m\}$ is present, the choice between a successful simulation or entering *exit* depends on the sequence of independent choices at state s_{t1} between the self-loop that receives f_S from c_S and the self-loop that receives m from c_M. As long as both these transitions are enabled, this choice gives equal probability to the two outcomes. Notice that we can view this as a stochastic variable with negative binomial distribution. Here, receiving a message from c_M is considered to be a success, while receiving a copy of f_S from c_S is a failure. Since, during each experiment, both outcomes have probability 0.5, it follows that, with probability 0.5, the operation will consume all k_M copies of m and one copy of a member of $M - \{m\}$ before consuming k_M messages from c_S. Hence with probability at least 0.5 the operation will lead to state s' with the value of c_S at least $k_S - k_M$. □

8 Correctness

We are now ready to state and prove the main theorem.

Theorem 17 *A control state s_F is reachable in $\mathcal{C}$ iff s_F is visited with probability one in $\mathcal{L}$.*

PROOF. $\Leftarrow$ **direction:** Assume that a control state is not reachable in $\mathcal{C}$. By Lemma 13, there is a positive probability that all simulations of send operations are faithful for the entire computation. It is clear that all rounds of such a computation correspond to computations in $\mathcal{C}$. This implies that the corresponding control state is not visited by $\mathcal{L}$ with a positive probability. This proves the theorem in this direction.

$\Rightarrow$ **direction:** Suppose that s_F is reachable in $\mathcal{C}$ by a computation σ of length n from the initial state. By Lemma 7 we know that, with probability one, a computation of $\mathcal{L}$ is of the form

$$\pi_1^1\pi_2^1\pi_2^1 \cdots \pi_{n_1}^1\pi_{rs}^1\pi_{cu}^1 \quad \pi_1^2\pi_2^2 \cdots \pi_{n_2}^2\pi_{rs}^2\pi_{cu}^2 \quad \cdots \quad \pi_1^i \cdots \pi_{n_i}^i\pi_{rs}^i\pi_{cu}^i \quad \cdots$$

Let K be any natural number such that

$$2^{K'} - 1 > n \cdot \left(K' \cdot (r+\alpha)^{n+1} + r\right)$$

for each $K' \geq K$, where α is defined as in Lemma 14. Obviously, such a K exists.

Let k_T^i be the value of c_T when entering π_{rs}^i, and let k_S^i be the value of c_S upon termination of the restart segment π_{rs}^i.

By Lemma 12 (b), with probability one, there is an infinite sequence $i_0 i_1 i_2, \ldots$ such that $k_T^{i_j} \geq K$ for each $j \geq 0$.

By Lemma 15 it follows that there is a positive ρ_1 such that, with probability ρ_1, we have $k_S^{i_j} \geq 2^{k_T^{i_j}} - 1$. Since $k_T^{i_j} \geq K$ it follows that $k_S^{i_j} > n \cdot \left(k_T^{i_j} \cdot (r+\alpha)^{n+1} + r\right)$. Consider the behaviour of $\mathcal{L}$ in the next round $i+1$.

By Lemma 14 the value of c_T will not exceed $k_T^{i_j} \cdot (r+\alpha)^{n+1} + r$ during the next n segments. By Lemma 16 and simple observations about the send, empty, and receive segments, the next n segments will lead to s_F with a probability which is bounded from below. The result follows immediately. $\Box$

From Theorem 17 and Theorem 2 we get the following.

Theorem 18 *The problem Reach-PLCS is undecidable.*

9 Conclusion

We have considered a model of probabilistic lossy channel systems (PLCSs) where each message my be lost during a send operation with a predefined probability. We show undecidability of the reachability problem for such a model.

On the other hand, [4] considers a slight different model; namely during the execution of the PLCS, there is a predefined probability by which the next step is a loss. The paper shows decidability assuming that the probability of losses is at least 0.5. The problem of whether the decidability result of [4] extends to the case where the probability of losses is less than 0.5 is still open.

The papers [3,5] consider yet a different model of PLCS, where each message inside a channel may be lost during each step of the PLCS. The papers show decidability of reachability regardless of the given probability of message losses.

Acknowledgments We are grateful to Marta Kwiatkowska for prompting this work by an invitation to PROBMIV 1999. Many thanks to Dag Jonsson for many helpful discussions on the theory of stochastic processes, and in particular for pointing out the potential applications of branching processes and processes with negative binomial distributions.

References

[1] Parosh Aziz Abdulla and Bengt Jonsson. Undecidable verification problems for programs with unreliable channels. *Information and Computation*, 130(1):71–90, 1996.

[2] Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[3] Parosh Aziz Abdulla and Alexander Rabinovich. Verification of probabilistic systems with faulty communication. In *Proc. FOSSACS03, Conf. on Foundations of Software Science and Computation Structures*, volume 2620, 2003.

[4] C. Baier and B. Engelen. Establishing qualitative properties for probabilistic lossy channel systems. In Katoen, editor, *ARTS'99, Formal Methods for Real-Time and Probabilistic Systems, 5th Int. AMAST Workshop*, volume 1601 of *Lecture Notes in Computer Science*, pages 34–52. Springer Verlag, 1999.

[5] N. Bertrand and Ph. Schnoebelen. Model checking lossy channels systems is probably decidable. In *Proc. FOSSACS03, Conf. on Foundations of Software Science and Computation Structures*, volume 2620, 2003.

[6] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 2(5):323–342, April 1983.

[7] M. H. DeGroot. *Probability and Statistics.* Addison-Wesley, 1975.

[8] Purush Iyer and Murali Narasimha. Probabilistic lossy channel systems. In *TAPSOFT '97: Theory and Practice of Software Development*, volume 1214 of *Lecture Notes in Computer Science*, pages 667–681, 1997.

[9] S. Karlin. *A First Course in Stochastic Processes.* Academic Press, 1966.

[10] J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains.* D Van Nostad Co., 1966.

[11] M.O. Rabin. Probabilistic algorithms. In J.F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 21–39. Academic Press, 1976.

Appendix – Proof of Lemmas

Proof of Lemma 10

Statement of the Lemma: Let X be a branching process with parameters r and p. If $r \cdot p > 2$ then there is a positive ρ such that for all i, the probability that $X_i \geq 2^i$ is greater than ρ.

PROOF. It is a simple property of binomial distributions that

$$E(X_1) = r \cdot p \qquad Var(X_1) = r \cdot p \cdot (1-p)$$

By results from the theory of branching processes [9] the expected value and variance of X_n are given by

$$\begin{aligned} E(X_n) &= (E(X_1))^n = (r \cdot p)^n \\ Var(X_n) &= Var(X_1) \cdot E(X_1)^{n-1} \cdot \tfrac{E(X_1)^n - 1}{E(X_1) - 1} \end{aligned}$$

We use *Chebyshevs' inequality*, which states that for any random variable Y and positive real K, we have

$$\mathcal{P}(|Y - E(Y)| \geq K) \leq \frac{Var(Y)}{K^2}$$

which implies that

$$\mathcal{P}(Y \geq E(Y) - K) \geq 1 - \frac{Var(Y)}{K^2}$$

We know that $E(X_n) > 2^n$. Take $K = E(X_n) - 2^n$. It follows that

$$\mathcal{P}(X_n \geq 2^n) \geq 1 - \frac{Var(X_n)}{(E(X_n) - 2^n)^2} = 1 - \frac{Var(X_n)}{E(X_n)^2 + 2^{2n} - E(X_n) \cdot 2^{n+1}}$$

Substituting the value of $E(X_n)$ and $Var(X_n)$ we get

$$\mathcal{P}(X_n \geq 2^n) \geq 1 - \tfrac{Var(X_1) \cdot E(X_1)^{n-1} \cdot (E(X_1)^n - 1)}{(E(X_1)-1) \cdot (E(X_1)^{2n} + 2^{2n} - E(X_1)^n \cdot 2^{n+1})} =$$
$$1 - \tfrac{Var(X_1)}{(E(X_1)-1)E(X_1)} \; \tfrac{E(X_1)^{2n-1} - E(X_1)^{n-1}}{(E(X_1)^{2n-1} + \frac{2^{2n}}{E(X_1)} - E(X_1)^{n-1} 2^{n+1})}$$

We observe that

$$\lim_{n \to \infty} \frac{E(X_1)^{2n-1} - E(X_1)^{n-1}}{(E(X_1)^{2n-1} + \frac{2^{2n}}{E(X_1)} - E(X_1)^{n-1} 2^{n+1})} \quad = \quad 1$$

This means that for any $\epsilon > 0$ there is an N such that for each $n \geq N$ we have

$$\frac{E(X_1)^{2n-1} - E(X_1)^{n-1}}{(E(X_1)^{2n-1} + \frac{2^{2n}}{E(X_1)} - E(X_1)^{n-1}2^{n+1})} \;\; \leq \;\; 1 + \epsilon$$

From the fact that $r \cdot p > 2$ and $1 - p < 1$, we can in particular choose ϵ such that

$$\epsilon < \frac{r \cdot p - 1}{1 - p} - 1$$

This implies that there is an N such that for each $n \geq N$ we have

$$\mathcal{P}(X_n \geq 2^n) \geq 1 - \frac{Var(X_1)}{(E(X_1)-1) \cdot E(X_1)} \cdot (1+\epsilon) = 1 - \frac{(1-p) \cdot E(X_1)}{(E(X_1)-1) \cdot E(X_1)} \cdot (1+\epsilon) = 1 - \frac{1-p}{r \cdot p - 1} \cdot (1+\epsilon) > 0$$

Let $\rho_i = \mathcal{P}(X_i \geq 2^i)$. Since $r \cdot p > 2$ (and consequently $r > 2$) it follows that $\rho_i > 0$ for each $i \geq 0$. Define

$$\rho \;=\; \min\left(\rho_0, \rho_1, \ldots, \rho_{N-1}, 1 - \frac{1-p}{r \cdot p - 1} \cdot (1+\epsilon)\right)$$

$\square$

Proof of Lemma 11

Statement of the Lemma: Consider $p : 0 < p < 1$. There is a natural number α and a positive rational ρ such that the following holds: Let $y > 0$ and let X be a stochastic variable which has a negative binomial distribution with parameters y and p. Then, the probability of $X \leq \alpha \cdot y$ is greater than ρ.

PROOF. We define $\alpha \geq \frac{q+1}{p}$ and $\rho = 1 - q$. We show that α and ρ satisfy the property stated in the Lemma. Let $y > 0$ and let X be a stochastic variable which has a negative binomial distribution with parameters y and p.

In [7] it is shown that

$$E(X) = \frac{y \cdot q}{p} \qquad Var(X) = \frac{y \cdot q}{p^2}$$

By Chebyshev's inequality we know that for any positive real K:

$$\mathcal{P}(|X - E(X)| \geq K) \leq \frac{Var(X)}{K^2}$$

which implies

$$\mathcal{P}(X \leq E(X) + K) \geq 1 - \frac{Var(X)}{K^2}$$

In particular if we take $K = \frac{y}{p}$ then

$$\mathcal{P}(X \leq E(X) + \frac{y}{p}) \geq 1 - \frac{p^2 \cdot Var(X)}{y^2}$$

Substituting the values of $E(X)$ and $Var(X)$ we get

$$\mathcal{P}(X \leq y \cdot \frac{q+1}{p}) \geq 1 - \frac{p^2 \cdot y \cdot q}{y^2 \cdot p^2} = 1 - \frac{q}{y} \geq 1 - q$$

The result follows from the definitions of α and ρ. $\square$