Specifying and Verifying the Alternating Bit Protocol with Probabilistic-Timed LOTOS *

Carlos Gregorio-Rodríguez and Manuel Núñez
Dept. de Informática y Automática
Universidad Complutense de Madrid
E-28040 Madrid, Spain
{cgregor,manuelnu}@dia.ucm.es

Abstract

In this paper we use a probabilistic-timed version of LOTOS *(*PTLOTOS*) for specifying and verifying the Alternating Bit Protocol (ABP). First, we present the language which is an upward compatible version of* LOTOS *where some new operators to deal with time and probabilistic features have been included. We give an operational semantics, where there are two kinds of transitions: probabilistic transitions and timed transitions. Transitions of the first kind are associated with the events that a behavior expression can execute, while timed transitions deal with time evolution. From this operational semantics we define a testing semantics, where a process passes a test with a probability in a period of time, and two processes are intended to be equivalent if they pass all the tests with the same probability for any time.*

In the second part of the paper, we use PTLOTOS *in order to specify the Alternating Bit Protocol. The ABP is a simple communication protocol which provides an error free communication over a faulty medium. The protocol has three components: the sender, the receiver and the (faulty) medium, where we suppose that the medium may lose the messages. The only observable events of the whole system will be* **mess** *(indicating that a message has been sent) and* **deli** *(indicating that a message has been received and delivered). Once we have specified the three components and we have described the whole system, we verify that our specification fulfills some good properties. Because of the testing semantics we have defined, we can make assumptions about the environment, and we can express them using tests, and so, we can get information about the behavior of the protocol by studying the interaction between such tests and the specification of the ABP.*

*Research supported in part by the CICYT project TIC 94-0851-C02-02

1 Introduction

The growing, both in extension and complexity, of the problems whose solutions have to be found with the help of computers make not only interesting but necessary the use of tools and techniques which assure the efficiency and verify the desired properties of the hardware and software under construction.

Formal methods settle a suitable theoretical mathematical background which must support these tools and techniques, besides warranting the correctness (soundness) of the results getting out by using such tools and techniques.

Concurrency is one of the nowadays subjects to deal with in computation which has increased in a qualitative way the complexity of the problems to be solved. Several formal methods have focused modeling concurrency from different levels of abstraction: Petri Nets, Process Algebras and Logics. We consider process algebras as the most fertile formalism because they can be useful in a wide and interesting spectrum, namely specification and design; furthermore, given that the abstraction level of these formalisms lies between that of Petri nets and that of logics, they are suitable to study the properties of the processes, without losing the internal structure, as logics do, and without taking into account many low level details, as Petri nets do.

LOTOS [15] is a Formal Description Technique based on process algebras and the formal theory of abstract data types. LOTOS is a FDT generally applicable to distributed, concurrent, and information processing systems, however, it has been developed for the formal specification of open distributed systems, and in particular for those related to the open systems interconnection (OSI).

LOTOS enjoys nice properties, inherited from the process algebras, such as abstraction, modularity and composition, but though we can express a wide range of properties that are relevant for the description of OSI services, protocols and interfaces, we cannot describe, clearly or directly, some probabilistic or timed behaviors. In fact, in order to describe communicating systems, it is often necessary to deal with probabilistic and statistical phenomena such as randomization or failure rates: for example, to specify the possibility of a failure or to express some kind of *fairness* in choices; on the other hand, time is an inherent quality of the nature of processes, so the much we may manage it the better it is: for instance, to express the exact period of time in which a process is willing to offer an action, or to be able to describe a delay.

The capabilities of a model to manage time and probability are not only worth for the expressiveness of the language, but also because they allow to study a wider and richer range of properties over the processes of the language.

In the last years there have been several proposals for the inclusion of timed information in LOTOS (e.g. [16, 11, 12]), using different approaches to introduce time; for example, in [12] (the improved version of [11]), an operator similar to that in [23] is used, denoting that an action is offered to the environment during a time interval, while in [16], actions are available only at a given point of time. Moreover, these formalisms have been assessed in the specifications of a timed protocol [13, 18]. These proposals have led to the current version of Timed LOTOS presented in [3], where LOTOS is extended with three time operators: a delay operator, a timed prefix operator and a timed choice. On the other hand, probabilistic extensions of LOTOS are not so many. For example, in [17, 9] LOTOS is extended with a probabilistic internal choice, while both of them keep the nonprobabilistic choice operator. Using a different style, in [21] the usual choice and parallel operators are extended with a probability parameter instead of adding a new probabilistic operator, so the probabilistic version of the language is closer to the nonprobabilistic one.

In this paper we present a probabilistic-timed extension of LOTOS, namely PTLOTOS. This language will be a conjunction of previous proposal for Timed LOTOS and Probabilistic LOTOS. In particular, we will consider the current timed extension of LOTOS described in [3], while the probabilistic part of the language will be based on the generative interpretation of probabilities described in [21].

The rest of the paper is structured as follows. In Section 2 we present the new language. We define an operational semantics, and from it we define a (probabilistic-timed) testing semantics. In this semantics, two behavior expressions are considered equivalent iff for every test the probability of passing a test until a given time is equal for both behavior expressions. Once presented the semantics of PTLOTOS, in Section 3 we present our case study. We specify the ABP and verify that our specification fulfills some good properties by testing the specification with some suitable tests. Finally, in Section 4 we present our conclusions and some lines for future work.

2 PTLOTOS: Probabilistic-Timed LOTOS

We will extend a subset of LOTOS with time and probabilistic features. This new extended LOTOS, namely PTLOTOS, will be based on the current timed version of LOTOS [3] and on the probabilistic version defined in [21]. Time will be introduced in the language by using a delay operator, $\mathrm{wait}(d)\,;B$, meaning that the behavior expression B will be *active* after a time d, and by extending the action prefix operator with a time interval, $e\{d_1..d_2\}\;;B$, meaning that the event e is available from time d_1 until time d_2. The differences with respect to [3] are two: we do not consider a time parameter t which can be *passed* to the behavior expression preceded by the prefix, and we do not allow the second time parameter associated with an internal event, i, to be equal to ∞, that is, we want to forbid the possibility of an internal event idling forever[1]. Probabilistic features are introduced by extending the choice and parallel operators with a probability which is used to assign *weights* to the components of the compositions.

By now, we consider neither *enabling* nor *disabling* operators, although we consider that it would not be difficult to include these operators in our language.

We will consider a universe of gates $\mathcal{G}$ that includes all the gate names, where $g, g', g_1, \dots$ range over $\mathcal{G}$, and $G, G', G_1, \dots$ range over sets of gate names. We will also consider an internal event i which is not visible for an external observer. The set $\mathcal{E} = \mathcal{G} \cup i$ is the universe of events, where $e, e', e_1, \dots$ range over $\mathcal{E}$. Regarding the timed part of the language, we consider a finite set Id of process variables. We consider a (dense) countable time domain, denoted by D, which is the alphabet of time actions (this set is formally defined in [3]), and we consider the subsets of the time domain $D_0 = D - \{0\}$, and $D_\infty = D - \{\infty\}$.

Definition 1 The set of PTLOTOS *behavior expressions* is defined by the following BNF-expression:

$$\begin{aligned} B ::= \; & \mathrm{stop} \mid X \mid \mathrm{wait}(d)\;;B \mid g\{d_1..d_2\}\;;B \mid \\ & i\{d_1..d_1'\}\;;B \mid B[\,]_p B \mid B\,|[G]|_p\,B \mid \\ & \mathrm{hide}\; G \;\mathrm{in}\; B \mid X := B \end{aligned}$$

where $p \in (0,1)$, $g \in \mathcal{G}$, $d, d_1, d_1' \in D_\infty$, $d_2 \in D$ such that $d_1 \le d_1', d_2$, $G \subseteq \mathcal{G}$ and $X \in Id$. □

[1] This last decision will be consistent with the internal events produced by a hide operator.

Probabilistic Transitions	Timed Transitions
	$(STOP)\ \dfrac{}{\text{stop}\stackrel{d}{\rightarrowtail}\text{stop}}$
$(PRE1)\ \dfrac{}{g\{0..d_2\};B\xrightarrow{g}_1 B}$	$(PRE2)\ \dfrac{}{g\{d_1{+}d..d_2{+}d\};B\stackrel{d}{\rightarrowtail}g\{d_1..d_2\};B}$
	$(PRE3)\ \dfrac{d>d_2}{g\{d_1..d_2\};B\stackrel{d}{\rightarrowtail}\text{stop}}$
	$(PRE4)\ \dfrac{d>d_1}{g\{d_1..d_2{+}d\};B\stackrel{d}{\rightarrowtail}g\{0..d_2\};B}$
$(INT1)\ \dfrac{}{i\{0..d_2\};B\xrightarrow{i}_1 B}$	$(INT2)\ \dfrac{}{i\{d_1{+}d..d_2{+}d\};B\stackrel{d}{\rightarrowtail}i\{d_1..d_2\};B}$
	$(INT3)\ \dfrac{d>d_1}{i\{d_1..d_2{+}d\};B\stackrel{d}{\rightarrowtail}i\{0..d_2\};B}$
$(DEL1)\ \dfrac{B\xrightarrow{e}_p B'}{\text{wait}(0);B\xrightarrow{e}_p B'}$	$(DEL2)\ \dfrac{}{\text{wait}(d'{+}d);B\stackrel{d}{\rightarrowtail}\text{wait}(d');B}$
	$(DEL3)\ \dfrac{B\stackrel{d}{\rightarrowtail}B'}{\text{wait}(d');B\stackrel{d+d'}{\rightarrowtail}B'}$

where $d \in D_0, d_1 \in D_\infty$ and $d_2, d' \in D$.

Figure 1: Operational rules of stop, action prefix and delay.

The choice operator can be generalized to manage an arbitrary number of arguments, and this generalization will be used in Section 3.

Definition 2 Let $B_1, B_2, \ldots, B_n$ be behavior expressions and $0 < p_1, p_2, \ldots, p_n < 1$ be such that $\sum p_i = 1$. We inductively define the behavior expression $\sum_{i=1}^{n}[p_i]B_i \quad (n \geq 2)$ as:

- $\sum_{i=1}^{2}[p_i]B_i = B_1[\,]_{p_1}B_2$
- $\sum_{i=1}^{n}[p_i]B_i = B_1[\,]_{p_1}\sum_{i=2}^{n}[\frac{p_i}{1-p_1}]B_i \qquad (n > 2)$

We usually will write $[p_1]B_1 + [p_2]B_2 + \cdots + [p_n]B_n$ instead of $\sum_{i=1}^{n}[p_i]B_i$. □

2.1 Operational Semantics of PTLotos

We will define the meaning of syntactic terms by means of an operational semantics. There will be two kinds of transitions: *probabilistic* transitions and *time* transitions. The intuitive meaning of a *probabilistic* transition $B \xrightarrow{e}_p B'$ is that if all the gate names are offered by the environment, then the probability with which the behavior expression B performs the event e and then behaves as B' is equal to p. The intuitive meaning of a *timed* transition $B \stackrel{d}{\rightarrowtail} B'$ is that the behavior expression B behaves as B' after an amount of time equal to d. This time is greater than zero.

In the following we use $B \not\longrightarrow$ as a shorthand for $\nexists e, p, B' : B \xrightarrow{e}_p B'$, $B \stackrel{e}{\not\longrightarrow}$ as a shorthand for $\nexists p, B' : B \xrightarrow{e}_p B'$, while $B \stackrel{d}{\not\rightarrowtail}$ will stand for $\nexists B' : B \stackrel{d}{\rightarrowtail} B'$.

The rules defining the operational semantics are given in Figures 1 and 2. These rules are presented using the style in [20], where timed transitions and probabilistic transitions are separated.

As in [21], we will not use indices which are sometimes used for distinguishing between the different derivations of the same probabilistic transition (see [5]). We will consider that we are working with multisets of probabilistic transitions instead of with set of transitions, and so we can control all the possible derivations of the same probabilistic transition. Another alternative is presented in [10], but it leads to an increment of the number of operational rules.

Example 1 Let $B = g\{0..1\}\,;\text{stop}[\,]_{\frac{1}{2}}g\{0..2\}\,;\text{stop}$. Then, the transition $B \xrightarrow{g}_{\frac{1}{2}} \text{stop}$ will appear two times. □

The operational rules for the stop, prefix and delay operators are given in Figure 1. The parameter of the action prefix operator indicates the interval of time in which the corresponding action is available. If this interval of time is *consumed* and the event prefixing is a gate name, then the behavior expression

Probabilistic Transitions

$$(CHO1)\ \frac{B_1 \xrightarrow{e}_q B_1'}{B_1 [\,]_p B_2 \xrightarrow{e}_{\frac{p\cdot q}{r(B_1,B_2,\emptyset,p)}} B_1'}$$

$$(CHO2)\ \frac{B_2 \xrightarrow{e}_q B_2'}{B_1 [\,]_p B_2 \xrightarrow{e}_{\frac{(1-p)\cdot q}{r(B_1,B_2,\emptyset,p)}} B_2'}$$

$$(PAR1)\ \frac{B_1 \xrightarrow{e}_q B_1' \wedge\ e \notin G}{B_1 |[G]|_p B_2 \xrightarrow{e}_{\frac{p\cdot q}{r(B_1,B_2,G,p)}} B_1' |[G]|_p B_2}$$

$$(PAR2)\ \frac{B_2 \xrightarrow{e}_q B_2' \wedge\ e \notin G}{B_1 |[G]|_p B_2 \xrightarrow{e}_{\frac{(1-p)\cdot q}{r(B_1,B_2,G,p)}} B_1 |[G]|_p B_2'}$$

$$(PAR3)\ \frac{B_1 \xrightarrow{g}_s B_1' \wedge\ B_2 \xrightarrow{g}_t B_2' \wedge\ g \in G}{B_1 |[G]|_p B_2 \xrightarrow{g}_{\frac{s\cdot t}{r(B_1,B_2,G,p)}} B_1' |[G]|_p B_2'}$$

$$(HID1)\ \frac{B \xrightarrow{e}_q B' \wedge\ e \notin G}{\text{hide } G \text{ in } B \xrightarrow{e}_q \text{hide } G \text{ in } B'}$$

$$(HID2)\ \frac{B \xrightarrow{g}_q B' \wedge\ g \in G}{\text{hide } G \text{ in } B \xrightarrow{i}_q \text{hide } G \text{ in } B'}$$

Timed Transitions

$$(CHO3)\ \frac{B_1 \overset{d}{\rightarrowtail} B_1' \wedge\ B_2 \overset{d}{\rightarrowtail} B_2'}{B_1 [\,]_p B_2 \overset{d}{\rightarrowtail} B_1' [\,]_p B_2'}$$

$$(PAR4)\ \frac{B_1 \overset{d}{\rightarrowtail} B_1' \wedge\ B_2 \overset{d}{\rightarrowtail} B_2'}{B_1 |[G]|_p B_2 \overset{d}{\rightarrowtail} B_1' |[G]|_p B_2'}$$

$$(HID3)\ \frac{B \overset{d}{\rightarrowtail} B' \wedge\ \forall g \in G\colon (B \overset{g}{\not\longrightarrow} \wedge\ \forall B''\ \forall d' < d\colon (B \overset{d'}{\rightarrowtail} B'' \Rightarrow B'' \overset{g}{\not\longrightarrow}))}{\text{hide } G \text{ in } B \overset{d}{\rightarrowtail} \text{hide } G \text{ in } B'}$$

$$(REC1)\ \frac{B \xrightarrow{e}_q B' \wedge\ X := B}{X \xrightarrow{e}_q B'} \qquad (REC2)\ \frac{B \overset{d}{\rightarrowtail} B' \wedge\ X := B}{X \overset{d}{\rightarrowtail} B'}$$

where $d \in D_0, d_1 \in D_\infty$ and $d_2, d' \in D$.

Figure 2: Operational rules of choice, parallel, hide and recursion.

$$
\begin{aligned}
r(B_1, B_2, G, p) &= p \cdot \sum_{B_1'} \{\!\mid q \mid \exists e \notin G : B_1 \xrightarrow{e}_q B_1' \mid\!\} + (1-p) \cdot \sum_{B_2'} \{\!\mid q \mid \exists e \notin G : B_2 \xrightarrow{e}_q B_2' \mid\!\} \\
&+ \sum_{B_1', B_2'} \{\!\mid s \cdot t \mid \exists g \in G : B_1 \xrightarrow{g}_s B_1' \ \wedge \ B_2 \xrightarrow{g}_t B_2' \mid\!\}
\end{aligned}
$$

Figure 3: Normalization Factor.

becomes stop (see Rule ($PRE3$)); if the event is an internal event, that is i, then this internal event must be executed during the interval of time. These rules are similar to those in Timed LOTOS, and they define the most important design decisions about the timed part of the model.

The rest of the operational rules are given in Figure 2. We use a *normalization factor*, as defined in [21], in the rules for the choice and parallel operators. Intuitively speaking, $r(B_1, B_2, G, p)$ computes the sum of the probabilities associated with the events which do not belong to G that B_1 (resp. B_2) may perform, multiplied by p (resp. $1-p$), and the product of the probabilities with which B_1 and B_2 may (simultaneously) perform events belonging to G. That is, the normalization factor computes the *whole* probability with which both behavior expressions may perform events, considering the possible restrictions imposed by the synchronization set, besides taking also into account the probability p. This factor is formally defined in Figure 3.

The normalization factor is used in the choice operator (Rules ($CHO1$) and ($CHO2$)) in order to determine whether one of the components may not perform any probabilistic transition; in this case, the whole probability is assigned to the other component. Moreover, the normalization factor has as parameter the empty set of gate names because all the events could be performed by both behavior expressions.

Something similar happens to ($PAR1$) and ($PAR2$), but considering as parameter of the normalization factor the synchronization set. ($PAR3$) considers the case when a synchronization event may be performed by both sides of the parallel operator. ($CHO3$) and ($PAR4$) force both components to evolve simultaneously by a timed transition in order to get a timed transition from the composition. The rest of the rules are similar to those in Timed LOTOS (let us note that we keep the *urgency* of hidden events by using the rule ($HID3$)).

Following standard methods, the operational semantics induces the corresponding notion of *probabilistic-timed labeled transition systems*. Because this method is essentially the same that the one described in [3], but considering probabilistic distribution functions (see [21]), we have preferred to omit this formalization. Moreover, even though our operational semantics has rules with negative premises, this does not cause any problem because a stratification in terms of [6] can be given, so the uniqueness of (probabilistic-timed) labeled transition systems is assured.

Thus, at the semantic level, a behavior expression may be identified with the (unique) probabilistic-timed labeled transition system defining its operational semantics.

2.2 Testing Semantics

Testing semantics was first introduced in [4, 8]. The intuition behind the theory is quite simple: the behavior of processes is, in essence, to offer and to accept communications. Then, we would like to say that two processes are behavior equivalent if any user cannot distinguish the processes when interacting with them.

The experiments carried out by the user can only infer knowledge of a process by virtue of interacting or communicating with it. So, a probabilistic timed test will be just a behavior expression but extending the set of events with a new event $ok \notin \mathcal{G}$ indicating *successful* termination of the experiment. As in [14], we consider that the successful termination event ok is urgent, that is, ok stands for $ok\{0..0\}$; stop. Intuitively speaking, if a test may execute the successful termination event, then it must be executed *as soon as possible*. We have the following rule:

$$
(SUC)\frac{}{ok \xrightarrow{ok}_1 \text{stop}}
$$

Given that this is the unique rule of this new process, we have that the event ok does not allow the passage of time.

The second step is to define the interaction between a behavior expression and a test. This interaction will be modeled by the parallel composition of the tested behavior expression and the test, taking as synchronization set the full set of gate names (i.e. the acceptance event ok is not included), and as associated probability $\frac{1}{2}$. The choice of this probability has been done to keep the symmetry between the internal events of the behavior expression and those of the test. This interaction will be represented by $B|T$, and is formally defined as:

$$B \,|\, T = \text{hide } \mathcal{G} \text{ in } (B \,|[\mathcal{G}]|_{\frac{1}{2}}\, T)$$

So, our definition will be similar to that in [21] but considering the urgency of internal events.

The next step consists in defining a function computing the probability with which a behavior expression *passes* a test before a given amount of time has been consumed. This function must be carefully defined; otherwise we can get strange results. We will illustrate the problems that could appear by using the next

Example 2 Let B = stop, $T = i\{0..1\}$; *ok*, and consider $S = B \mid T$. The transitions that S may perform are of the following form:

$$S \overset{d_1}{\rightarrowtail} S_{d_1} = (\text{stop} \mid i\{0..1-d_1\} \,;\, ok)$$
$$S \xrightarrow{i}_1 (\text{stop} \mid ok) \xrightarrow{ok}_1 \text{stop} \mid \text{stop}$$

for any $0 < d_1 \leq 1$.

Now, for every d_1, S_{d_1} may perform the following transitions:

$$S_{d_1} \overset{d_2}{\rightarrowtail} S_{d_1+d_2} = (\text{stop} \mid i\{0..1-d_1-d_2\} \,;\, ok)$$
$$S_{d_1} \xrightarrow{i}_1 (\text{stop} \mid ok) \xrightarrow{ok}_1 \text{stop} \mid \text{stop}$$

for any $0 < d_2 \leq 1 - d_1$, and so on.

So the *successful computations*, that is, computations performing the event *ok* (this concept will be formally defined below), from S are infinite. In fact, there are infinite successful computations for any point of time in the interval (0,1]. Note that at time 0 there exists a unique successful computation:

$$S \xrightarrow{i}_1 (\text{stop} \mid ok) \xrightarrow{ok}_1 \text{stop} \mid \text{stop}$$

For instance, at time 0.5 we have (among others) the successful computations of the form:

$$S \overset{\frac{1}{2}}{\rightarrowtail} (\text{stop} \mid i\{0..\tfrac{1}{2}\} \,;\, ok) \xrightarrow{i}_1 (\text{stop} \mid ok) \xrightarrow{ok}_1 \text{stop} \mid \text{stop}$$
$$S \overset{\frac{1}{4}}{\rightarrowtail} (\text{stop} \mid i\{0..\tfrac{3}{4}\} \,;\, ok) \overset{\frac{1}{4}}{\rightarrowtail} (\text{stop} \mid i\{0..\tfrac{1}{2}\} \,;\, ok) \xrightarrow{i}_1 (\text{stop} \mid ok) \xrightarrow{ok}_1 \text{stop} \mid \text{stop}$$
$$S \overset{\frac{1}{8}}{\rightarrowtail} (\text{stop} \mid i\{0..\tfrac{7}{8}\} \,;\, ok) \overset{\frac{3}{8}}{\rightarrowtail} (\text{stop} \mid i\{0..\tfrac{1}{2}\} \,;\, ok) \xrightarrow{i}_1 (\text{stop} \mid ok) \xrightarrow{ok}_1 \text{stop} \mid \text{stop}$$

So, if we would consider that the probability with which B passes the test T before time 0.5 is equal to the sum of all the possible computations consuming at most time 0.5, we get that this probability is equal to ∞, because the probability associated with any of these transitions is equal to 1. □

The previous example shows that we cannot consider *all* these computations as successful ones. One possible solution (actually, the easiest one) would be to consider that the internal event i is urgent, that is, i does not allow the passage of time. But this goes against the current version of Timed LOTOS where $i\{d_1..d_2\}$ can be executed at any point of time between d_1 and d_2. Our choice has been to consider the passage of time as *prior* in the sense that in order to calculate successful computations we will fix our attention in those computations which have the maximal possible passage of time in any step. For instance, in the previous example we have that the probability with which the behavior expression B passes the test T is equal to 1, at time 1, because the unique (maximal in time for every timed transition) successful computation is

$$S \overset{1}{\rightarrowtail} (\text{stop}|i\{0..0\};ok) \xrightarrow{i}_1 (\text{stop}|ok) \xrightarrow{ok}_1 \text{stop}|\text{stop}$$

Definition 3 Let B be a behavior expression and T be a test. We call *computations* the (possibly infinite) sequences of transitions from $S_0 = B \mid T$ of the form

$$C = S_0 \overset{d_1}{\rightarrowtail} S_1' \xrightarrow{e_1}_{p_1} S_1 \overset{d_2}{\rightarrowtail} S_2' \xrightarrow{e_2}_{p_2} S_2 \ldots$$

where for any j, and for all $d > d_j$ we have $S_{j-1} \not\overset{d}{\rightarrowtail}$. As a convention, if any of the d_j is equal to zero, then S_{j-1} and S_j' must be the same (syntactic) test application (let us remember that zero timed transitions are not allowed). In fact, $S \overset{0}{\rightarrowtail} S$ means that for every $d \in D_0$ we have $S \not\overset{d}{\rightarrowtail}$. Note that for all j, $e_j = i$ or $e_j = ok$.

We distinguish *infinite computations* and *blocked computations*, which are those finite computations

$$C = S_0 \overset{d_1}{\rightarrowtail} S_1' \xrightarrow{e_1}_{p_1} S_1 \overset{d_2}{\rightarrowtail} S_2' \xrightarrow{e_2}_{p_2} S_2 \ldots \overset{d_n}{\rightarrowtail} S_n' \xrightarrow{e_n}_{p_n} S_n$$

that cannot be extended by a probabilistic transition, that is, $S_n \overset{d}{\rightarrowtail} S' \xrightarrow{e}_p S''$ for no d, S', e, p, S''. In this case, we will say that C has *length* n.

Given a blocked computation C, we inductively define its probability, $Pr(C)$, as

$$Pr(C) = \begin{cases} p & \text{if } \mathit{length}(C) = 1 \\ & \wedge\, C \overset{d}{\rightarrowtail} C'' \xrightarrow{e}_p C' \\ p \cdot Pr(C') & \text{if } \mathit{length}(C) > 1 \\ & \wedge\, C \overset{d}{\rightarrowtail} C'' \xrightarrow{e}_p C' \end{cases}$$

Given a blocked computation C, we inductively define its time, $T(C)$, as

$$T(C) = \begin{cases} d & \text{if } \mathit{length}(C) = 1 \\ & \vee\, C \overset{d}{\rightarrowtail} C'' \xrightarrow{ok}_p C' \\ d + T(C') & \text{if } \mathit{length}(C) > 1 \\ & \wedge\, C \overset{d}{\rightarrowtail} C'' \xrightarrow{i}_p C' \end{cases}$$

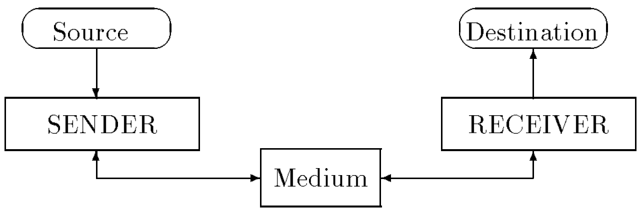

Figure 4: Communications System.

We call *successful computations* those blocked computations of S such that there exists some $j \in \{1, \ldots, n\}$ such that $e_j = ok$. We will denote by $\widetilde{S}$ the *set of successful computations of* S. □

Definition 4 Let B be a behavior expression and T be a test, and consider $S = B\,|\,T$. We say that B *pass* T with a *probability* equal to p *until time* d, denoted by $B\ pass_p^d\ T$, if

$$\sum_{\substack{C \in \widetilde{S} \\ T(C) \leq d}} Pr(C) = p$$

□

The previous definition has to be read as: d is the maximum amount of time needed to pass the test with a probability equal to p. We can assure that d is the maximum time because of the time restriction imposed in Definition 3 to computations. The previous definition induces the corresponding notion of probabilistic-timed testing equivalence.

Definition 5 (Testing Equivalence)
Given two behavior expressions B_1 and B_2, we write $B_1 \approx_T B_2$ iff for all tests T it holds:

$$\forall d \in D: \ B_1\ pass_p^d\ T \iff B_2\ pass_p^d\ T$$

□

Before finishing this section, let us remark again that tests and behavior expressions have the same syntax (except for the event *ok*). So, we are able to express the properties we would like to check out or the assumptions about the environment using a formal language: PTLotos.

3 Alternating Bit Protocol

The *Alternating Bit Protocol* [1] is a simplified representative of a class of protocols assuring the secure transmission of data over faulty media. This protocol has been previously specified using other models of concurrent processes. For example, using CCS appears in [19] and using timed or probabilistic models we can find it in [2, 22, 7] (for probabilistic processes, Timed CSP, and a Probabilistic-Timed version of CCS, respectively).

In this section we describe our specification of the ABP using PTLotos and comment the chosen design decisions. Then, we prove that the specification verifies some good properties by means of: first, describing the test which gathers the desired property; second, calculating the interaction system ABP|Test; and third, inferring the results from the interaction system.

The informal description of the protocol is as follows: the source produces messages which have to be delivered to the destination. As the transmission takes place over an unreliable medium, the sender and the receiver are needed to overcome this problem. The system is described in Figure 4.

We consider the *source* and the *destination* as the *environment*. The source communicates with the sender through the gate name **mess**, while the receiver and the destination share the gate name **deli**.

The sender behaves patiently ($\mathbf{mess}\{0..\infty\}$) waiting for the source to communicate with it; after receiving a message (indicated by the event **mess**), the sender sends the message to the medium and waits for an acknowledgment from the receiver. The period of time the sender is waiting for an answer from the medium is an arbitrary design decision which does not spoil the general behavior of the ABP: in this case we have chosen two time units. If the acknowledgment had not arrived before two time units were consumed, then the sender supposes that the message has been lost, and so, resends the message. The *duplication of states* and the one bit associated with messages and acknowledgments are necessary to avoid delivering duplicated messages and confusing current from previous acknowledgments. The specification of the sender follows:

$$\begin{array}{rcl}
S_1 & = & \mathbf{mess}\{0..\infty\}\,;S_2 \\
S_2 & = & sm_0\{0..\infty\}\,;S_3 \\
S_3 & = & [\frac{1}{3}]\ ra_0\{0..2\}\,;S_1' + [\frac{1}{3}]\ ra_1\{0..2\}\,;S_3 \\
 & + & [\frac{1}{3}]\ i\{2..2\}\,;S_2
\end{array}$$

$$\begin{array}{rcl}
S_1' & = & \mathbf{mess}\{0..\infty\}\,;S_2' \\
S_2' & = & sm_1\{0..\infty\}\,;S_3' \\
S_3' & = & [\frac{1}{3}]\ ra_1\{0..2\}\,;S_1 + [\frac{1}{3}]\ ra_0\{0..2\}\,;S_3' \\
 & + & [\frac{1}{3}]\ i\{2..2\}\,;S_2'
\end{array}$$

The receiver waits patiently a message signal from the medium. If the message is *fresh*, then it delivers it to the destination and sends an acknowledgment to the sender. If, on the contrary, the message is an old one, then it sends an acknowledgment to the sender and waits for the new message. The specification of the receiver follows:

$$
\begin{array}{rcl}
R_1 & = & [\frac{1}{2}]\ rm_0\{0..\infty\}\ ;\ R_2 + [\frac{1}{2}]\ rm_1\{0..\infty\}\ ;\ R'_3 \\
R_2 & = & \mathbf{deli}\{0..\infty\}\ ;\ R_3 \\
R_3 & = & sa_0\{0..\infty\}\ ;\ R'_1 \\
R'_1 & = & [\frac{1}{2}]\ rm_0\{0..\infty\}\ ;\ R_3 + [\frac{1}{2}]\ rm_1\{0..\infty\}\ ;\ R'_2 \\
R'_2 & = & \mathbf{deli}\{0..\infty\}\ ;\ R'_3 \\
R'_3 & = & sa_1\{0..\infty\}\ ;\ R_1
\end{array}
$$

In its initial state, the medium stays in a patience state waiting for a request from the sender or from the receiver. When a request is received, the medium, which is faulty, decides instantaneously either going on with the request or losing it. This decision is modeled in the states $M_2 \ldots M_5$ with the choice between the corresponding action and the i event, each one of these choices has a probability associated with the event i which corresponds with the probability of fault. In this case, we have chosen a medium losing 20% of the messages. The specification of the medium follows:

$$
\begin{array}{rcl}
M_1 & = & [\frac{1}{4}]\ sm_0\{0..\infty\}\ ;\ M_2 + [\frac{1}{4}]\ sa_0\{0..\infty\}\ ;\ M_3 \\
 & + & [\frac{1}{4}]\ sm_1\{0..\infty\}\ ;\ M_4 + [\frac{1}{4}]\ sa_1\{0..\infty\}\ ;\ M_5 \\
M_2 & = & [\frac{4}{5}]\ rm_0\{0..0\}\ ;\ M_1 + [\frac{1}{5}]\ i\{0..0\}\ ;\ M_1 \\
M_3 & = & [\frac{4}{5}]\ ra_0\{0..0\}\ ;\ M_1 + [\frac{1}{5}]\ i\{0..0\}\ ;\ M_1 \\
M_4 & = & [\frac{4}{5}]\ rm_1\{0..0\}\ ;\ M_1 + [\frac{1}{5}]\ i\{0..0\}\ ;\ M_1 \\
M_5 & = & [\frac{4}{5}]\ ra_1\{0..0\}\ ;\ M_1 + [\frac{1}{5}]\ i\{0..0\}\ ;\ M_1
\end{array}
$$

The complete system is described by the parallel composition of the initial states of the receiver, the sender and the medium. There is no communication between receiver and sender, so the first stage of the system must be

$$S_1\ |[\,]|_{\frac{1}{2}}\ R_1$$

On the other hand, all the gate names but **mess** and **deli** are communication actions between the medium and the previous subsystem, so they must belong to the synchronization set. Moreover, these are *internal* events of the system, so they must be hidden. Finally, we obtain

$$\mathbf{ABP} = \text{hide } A \text{ in } ((S_1\ |[\,]|_{\frac{1}{2}}\ R_1)\ |[A]|_{\frac{1}{2}}\ M_1)$$

where $A = \{sm_j, ra_j, rm_j, sa_j \mid j \in \{0,1\}\}$.

This system is shown in Figures 5 and 6. For the sake of clarity, we have preferred to indicate the hidden events (that is, those belonging to A) by their own names, but let us remark that all the events but **mess** and **deli** appearing in Figures 5 and 6 should be equal to i. Moreover, B^{-d} means a passage of time equals to d (this is similar to the predicate `Age` of Timed LOTOS).

3.1 Verification of Properties

In this section we deal with the verification of the ABP. One of the first properties to be proved is that every sent message has to be received. Let us suppose an *eager of knowledge* environment, that is, the source and destination are always ready to send and receive a message, respectively. Using the PTLOTOS syntax, this can be modeled by the test:

$$T_0 = \mathbf{mess}\{0..\infty\}\ ;\ \mathbf{deli}\{0..\infty\}\ ;\ ok$$

The interaction system $\mathbf{ABP}|T_0$ is shown in Figure 7. From this diagram we can infer, for instance, what is the maximum amount of time needed to assure that the message has been delivered with a probability greater than p. It is very easy to find the successful computations, and calculate their time and probability, from the diagram. Below, we show a table which has three entries: t stands for a (punctual) time, p_1 stands for the probability of the computations whose time is exactly t, and finally, p_2 gives the probability of passing the test T_0 before time t, that is, it is the cumulative sum of the previous probabilities.

$t \in$	$[0,2)$	$[2,4)$	$[4,6)$	$[6,8)$	$[8,10)$	$\cdots$
p_1	0.666	0.222	0.074	0.024	0.008	$\cdots$
p_2	0.666	0.888	0.962	0.987	0.995	$\cdots$

For example, under the previous design decisions, 4 time units are necessary to assure that the message arrives to the Destination with a probability greater than 0.9. In general, we can express the probability in function of the time: $\mathbf{ABP}\ pass_p^t\ T_0$ holds if

$$p = \sum_{i=0}^{d} \frac{1}{3^i} \cdot \frac{2}{3} \quad \text{where} \quad 2d \le t < 2(d+1)$$

In particular, if we consider $t = \infty$ we get $p = 1$, that is, we get that the message is delivered for sure.

By testing the ABP using the test T_0 we know the delay between the arrival of the message and its delivery, but we cannot know if the sender is willing again to receive another message from the source, or in other words, the sender does not know whether the previous message has arrived to the destination. We consider again an *ideal* patient environment, always waiting to interact with the ABP, which is modeled by using the test:

$$T_1 = \mathbf{mess}\{0..\infty\}\ ;\ \mathbf{deli}\{0..\infty\}\ ;\mathbf{mess}\{0..\infty\}\ ;\ ok$$

Figure 8 shows the interaction system ABP $| T_1$. As before, we can build the table below from the diagram.

$t \in$	$[0,2)$	$[2,4)$	$[4,6)$	$[6,8)$	$[8,10)$	$\cdots$
p_1	0.444	0.246	0.123	0.058	0.031	$\cdots$
p_2	0.444	0.691	0.814	0.873	0.905	$\cdots$

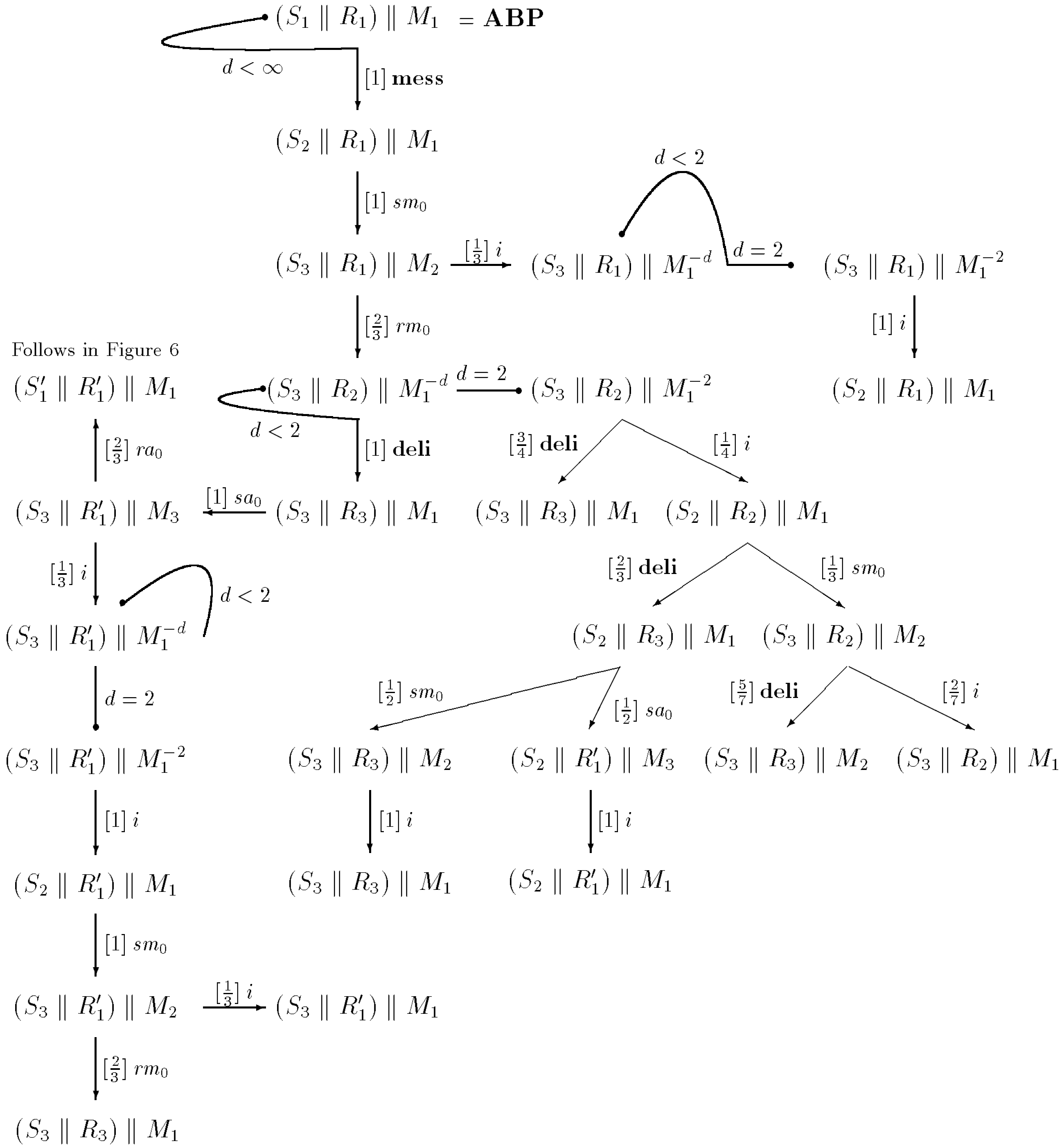

Figure 5: The Alternating Bit Protocol (1/2).

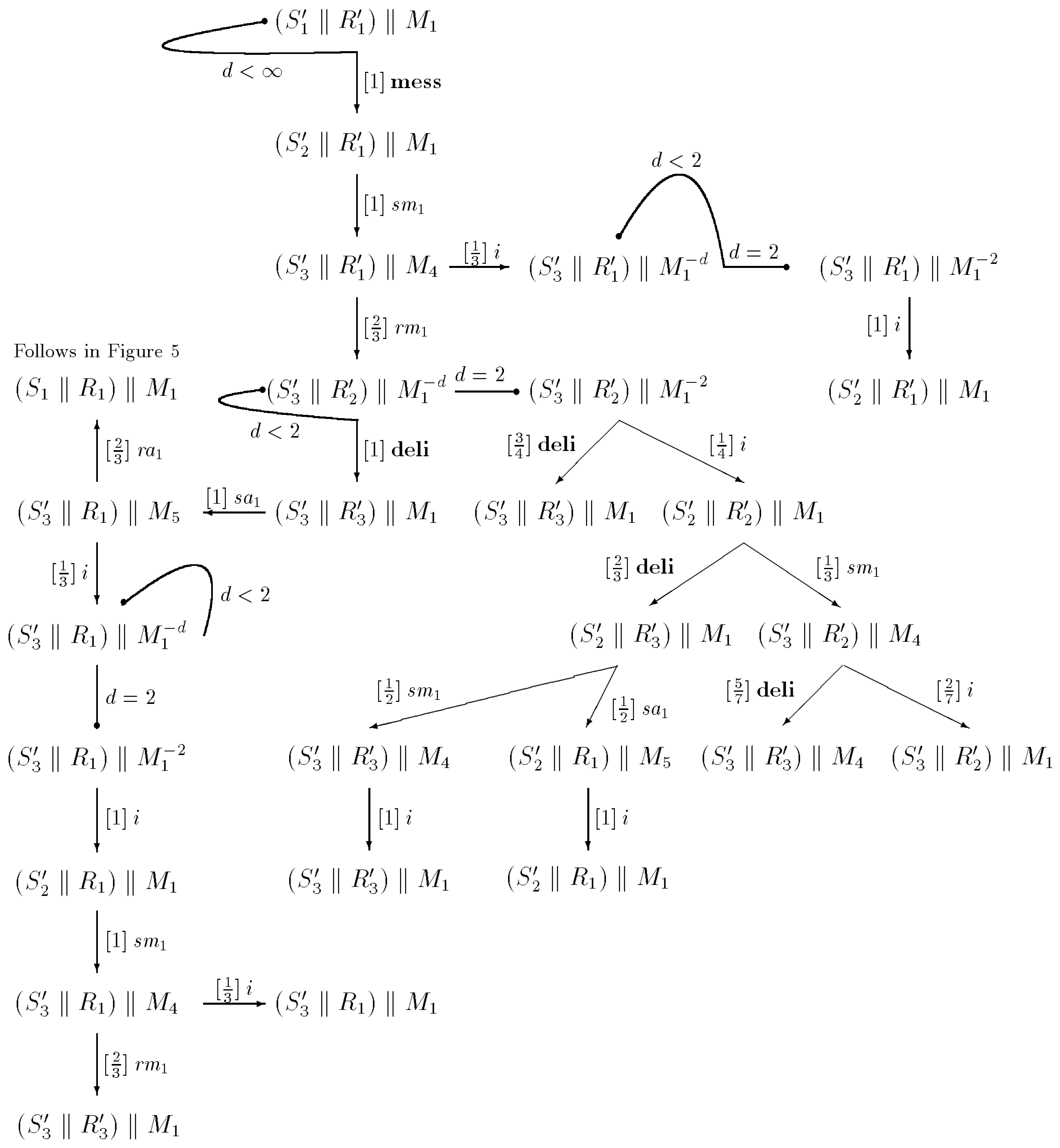

Figure 6: The Alternating Bit Protocol (2/2).

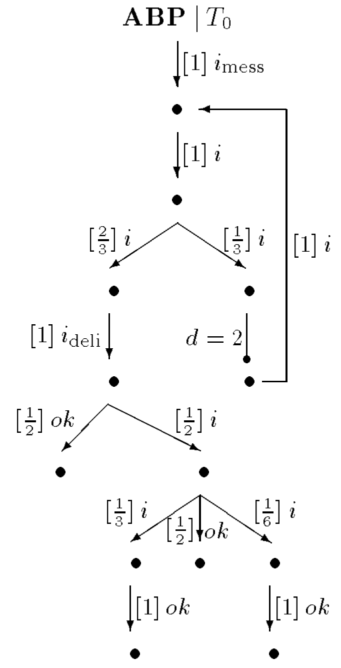

Figure 7: **ABP** | **mess**$\{0..0\}$; **deli**$\{0..0\}$; *ok*.

For example, under the previous design decisions, 8 time units are necessary to assure that the message arrives to the Destination with a probability greater than 0.9.

The table reveals a longer delay of time to achieve the same probability. This is because in the case of T_0, we finished the experiment after delivering the message without taking care of the fact that the medium could lose the acknowledgment which must be received by the sender in order to transmit more data. To get out the formula relating time and probability is not so easy as before, and cumbersome algebraic manipulations are needed. ABP $pass_p^t$ T_1 holds if

$$p = \sum_{k=0}^{d} \sum_{i+j=k} P_{l1}(i) \cdot P_{l2}(j)$$

where $2d \leq t < 2(d+1)$ and $P_{l1}(i)$ returns the sum of probabilities of the computations which spend exactly i time units in the *loop 1* and finish in the state in which the *loop 2* begins. On the other hand, $P_{l2}(j)$ returns the sum of probabilities of the *partial* computations which spend exactly j time units in the *loop 2*.

The function $P_{l1}(n)$ is easy to calculate: $\frac{1}{3^n} \cdot \frac{2}{3}$. On the contrary, the function $P_{l2}(n)$ is not trivial, because of the *little* delay loop inside of *loop 2*. The

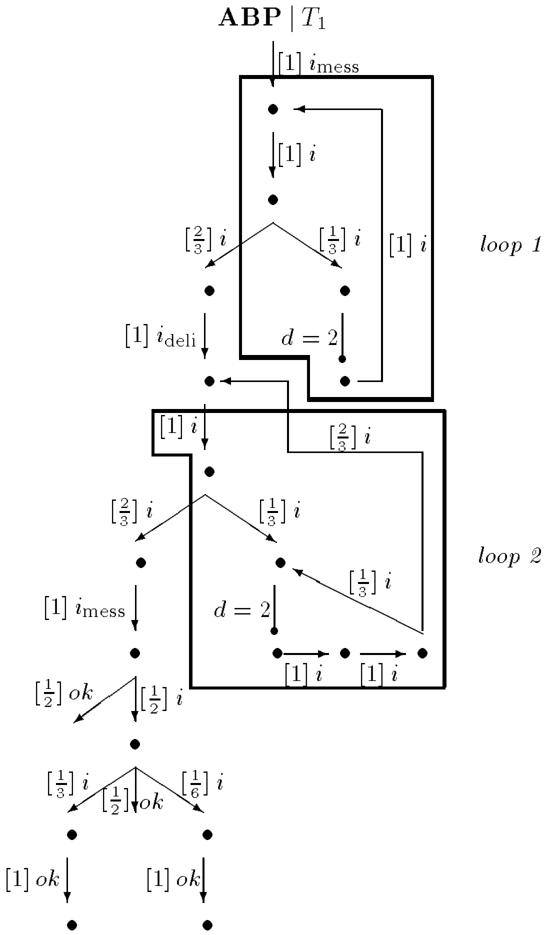

Figure 8: **ABP** |**mess**$\{0..0\}$;**deli**$\{0..0\}$;**mess**$\{0..0\}$;*ok*.

formula follows:

$$\begin{array}{rcll}
P_{l2}(n) & = & \sum_{k=0}^{n} P_{l2}^{k}(n) & \\
P_{l2}^{k}(n) & = & \sum_{d=0}^{n} \frac{2}{3} \frac{1}{3^{(d+1)}} P_{l2}^{k-1}(n-d-1) & \\
P_{l2}^{0}(n) & = & 0 & n > 0 \\
P_{l2}^{k}(0) & = & 0 & k > 0 \\
P_{l2}^{0}(0) & = & \frac{2}{3} &
\end{array}$$

Below, we show a table with some results of the function $P_{l2}^{k}(n)$.

$n \backslash k$	0	1	2	3	4
0	$\frac{2}{3}$	0	0	0	0
1	0	$\frac{2^2}{3^3}$	0	0	0
2	0	$\frac{2^2}{3^4}$	$\frac{2^3}{3^5}$	0	0
3	0	$\frac{2^2}{3^5}$	$\frac{2 \cdot 2^3}{3^6}$	$\frac{2^4}{3^7}$	0
4	0	$\frac{2^2}{3^6}$	$\frac{3 \cdot 2^3}{3^7}$	$\frac{2^5+2^4}{3^8}$	$\frac{2^5}{3^9}$

Like in the previous case, if we consider $t = \infty$, we get $p = 1$, that is, we get that the message is delivered for sure. The proof is done by showing that the probability of infinite idle is equal to zero.

Until now, we have considered quite simple tests but they do not have to be so simple. We can include probabilistic and time information into tests. Let us suppose that, unlike in the previous cases, the environment is not always ready to interact with the system. A simple way to express this idea is to consider that the environment expends some amount of time after sending the message up to being ready to deliver it. This can be modeled by the following test:

$$T_2 = \mathbf{mess}\{0..\infty\}\,;\mathrm{wait}(1)\,;\mathbf{deli}\{0..\infty\}\,;ok$$

Following the previous idea, that is, the environment is not patient, a more real environment could be one checking regularly, for a short period of time, if the message can be delivered, and if after some amount of time the message is not delivered, then the environment engages in internal activity for a period of time less than or equal to d. This quite *real* behavior can be described by means of the test that follows:

$$\begin{array}{lcl} T_3 & = & \mathbf{mess}\{0..\infty\}\,;T' \\ T' & = & (\mathbf{deli}\{0..1\}\,;ok)[\,]_{\frac{1}{2}}(i\{1..d+1\}\,;T') \end{array}$$

Like with the previous examples, to build up the interaction system is not very difficult, by using Figures 5 and 6 which show all the states of the ABP. To calculate tables of time and probability is routine but hard, because of the growing number of computations. There appear two solutions to overcome the problem: the first one consists in finding out the right algebraic laws to obtain a general formula, while the second one, actually the most practical, would be to use some suitable tools, such as tracers, in order to extract exhaustively all the successful computations.

4 Conclusions

In this paper we have presented a probabilistic-timed version of LOTOS. We have defined an operational semantics for the new language. From this operational semantics we have defined a testing semantics, where a behavior expression passes a test with a probability equal to p until time d if the sum of the probabilities associated with the successful computations of the interaction system between the behavior expression and the test consuming a time less than d is equal to p. Using the new language we have specified the *Alternating Bit Protocol*, and we have verified some expected properties. For instance, we get that if we allow an infinite amount of time, then the message will be delivered for sure.

While the ABP is not a complex protocol, we think that it suffices to notice that the new operators with which we extend LOTOS to manage time and probabilities are both intuitive and powerful, therefore PTLOTOS can also be useful for the specification of more complex systems which strongly depend on time information and random behaviors. By means of the testing semantics, we also provide a method for the verification of properties of such specifications.

As future work we plan on the one hand to define an alternative characterizations of our testing semantics by defining a notion of normal form, and a fully abstract denotational semantics, while on the other hand to build a tool performing all the computations that we had to perform *by hand*.

Acknowledgements

We would like to thank the (anonymous) proofreader of the paper for the useful corrections proposed.

References

[1] K.A. Bartlett, R.A. Scantlebury, and P.T. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Communications of the ACM*, 12(5):260–261, 1969.

[2] I. Christoff. *Testing Equivalences for Probabilistic Processes*. PhD thesis, Department of Computer Systems. Uppsala University, 1990.

[3] D. de Frutos, G. Leduc, L. Léonard, L. Llana, C. Miguel, J. Quemada, and G. Rabay. Time extended LOTOS. In *Working Draft on Enhancements to LOTOS*. ISO/IEC JTC1/SC21/WG1, 1995.

[4] R. de Nicola and M.C.B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[5] A. Giacalone, C.-C. Jou, and S.A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of Working Conference on Programming Concepts and Methods, IFIP TC 2*, 1990.

[6] Jan Friso Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118:263–299, 1993.

[7] H.A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. PhD thesis, Department of Computer Systems. Uppsala University, 1991.

[8] M. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.

[9] J.P. Katoen, R. Langerak, and D. Latella. Modeling systems by probabilistic process algebra: An event structures approach. In *Formal Description Techniques VI*, 1994.

[10] K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[11] G. Leduc and L. Léonard. A timed LOTOS supporting a dense time domain and including new timed operators. In *Formal Description Techniques V*, pages 87–102, 1992.

[12] L. Léonard and G. Leduc. An enhanced version of timed LOTOS and its application to a case study. In *Formal Description Techniques VI*, pages 483–498, 1993.

[13] L. Léonard, G. Leduc, and A. Danthine. The Tick-tock case study for the assesment of timed FDTs. In *The OSI95 Transport Service with Multimedia Support*. Research Reports ESPRIT - Project 5341 - OSI95, 1994.

[14] L. Llana, D. de Frutos, and M. Núñez. Testing semantics for urgent timed algebras. In *3rd AMAST Workshop on Real-Time Systems*, 1996.

[15] LOTOS. A formal description technique based on the temporal ordering of observational behaviour. IS 8807, TC97/SC21, 1988.

[16] C. Miguel, A. Fernández, and L. Vidaller. Extending LOTOS towards performance evaluation. In *Formal Description Techniques V*, pages 103–118, 1992.

[17] C. Miguel, A. Fernández, and L. Vidaller. LOTOS extended with probabilistic behaviours. *Formal Aspects of Computing*, 5:253–281, 1993.

[18] C. Miguel, A. Fernández, and L. Vidaller. Assesment of extended LOTOS. In *The OSI95 Transport Service with Multimedia Support*. Research Reports ESPRIT - Project 5341 - OSI95, 1994.

[19] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[20] F. Moller and C. Tofts. A temporal calculus of communicating systems. In *CONCUR'90, LNCS 458*, pages 401–415, 1990.

[21] M. Núñez and D. de Frutos. Testing semantics for probabilistic LOTOS. In *Formal Description Techniques VIII*, 1995.

[22] S. Schneider. *Correctness and Communication in Real-Time Systems*. PhD thesis, Oxford University, 1989.

[23] W. Yi. CCS+ Time = an interleaving model for real time systems. In *18th ICALP, LNCS 510*, pages 217–228, 1991.