Obtaining Probabilistic Dynamic State Graphs for TPPAL processes *

Juan J. Pardo Valentín Valero M. Carmen Ruiz Fernando Cuartero Gregorio Díaz
Computer Science Department
Esc. Politécnica Superior de Albacete, University of Castilla-La Mancha
Campus Universitario s/n, 02071 Albacete, Spain
{jpardo,valentin,carmenr,fernando,gregorio}@info-ab.uclm.es

Abstract

Software engineers work gladly with process algebras, as they are very similar to programming languages. However, graphical models are better in order to understand how a system behaves, and even these graphical models allow us to analyze some properties of the systems. Then, in this paper we present two formalisms for the specification of concurrent systems. On the one hand we present the timed-probabilistic process algebra TPPAL, which is a suitable model for description of systems in which time and probabilities are two important factors to be considered in the description, as it occurs in real-time systems and fault-tolerant systems. Then, the specification written in TPPAL can be automatically translated into a graphical model (the so-called probabilistic dynamic state graphs), which allows us to simulate and evaluate the system. Thus, in this paper we present this translation, which is currently supported by a tool (TPAL).

1 Introduction

Process algebras have been introduced to specify the behavior of systems, due to their simplicity and conciseness. In this way, the system is described by using a finite set of operators, which brings on specifications more readable, simpler and easier to modify than other classical notations, such as automata.

We can find a wide range of systems for which time becomes a very important factor to be considered in the specifications (real-time systems, fault-tolerant systems, etc.). These systems have some common characteristics: first of all, we have to associate time with actions, but we also may find some critical actions, which must be executed urgently, once they become enabled. In the last years, therefore, several extensions of the classical models have been defined in order to incorporate these factors: we have timed extensions of CSP [RR88, Ort90, Sch95], CCS [MT90], and LOTOS [QAdF89], and [BL91] where the authors include an operator for the specification of urgent interactions. There are some other proposals, which combine both factors, time and probabilities, concretely [HJ90, Han92] are extension of CCS.

Other extensions add probabilistic information to the classical models so we can quantify some decisions whereas other decisions are non-deterministic. The first reference which includes probabilities into process algebras is [LS89]. [CdFV97] presents a probabilistic extension of CSP where the authors change the internal choice for a probabilistic generative choice and they change the external choice for a probabilistic reactive one. Another probabilistic extension of CSP was presented in [Nuñ03].

Some researchers have proposed some other models which include both time and probabilities [And00, BA03].

The model we consider (TPPAL) has been defined on the basis of the classical operators of basic LOTOS. We have maintained the external choice, the parallel composition, and the hiding and recursion operators, and we have added some new operators for the specification of timed behaviors and urgent interactions; we have also added a probabilistic choice operator.

Concretely, according to the ideas of [QAdF89], we introduce a timed prefix operator, $a < t_1, t_2 >; P$, which establishes a time interval for the execution of the corresponding action, but once the deadline of the action has expired it cannot be executed ever after. Other operators we include are: an internal action prefix, $\tau; P$, which establishes the urgent character of the τ action, and a delay operator, $wait(t); P$, which establishes a delay of t units of time, and then it behaves

*This work has been supported by the CICYT project "Description and Evaluation of Distributed System and Application to Multimedia System",TIC2003-07848-C02-02.

as $\tau; P$. We also add a probabilistic choice operator $\sum_{i \in I} [q_i] N_i$.

We define an operational semantics for the language, which captures the aforementioned behavior, and particularly the urgent character of the internal action. We also define a characterization of TPPAL processes by means of a kind of probabilistic dynamic state graphs, which allows us to simulate the execution of TPPAL processes as well as to analyze their behavior.

The paper is structured as follows. In Section 2 we present the syntax and semantics of the language TPPAL, while the model of probabilistic dynamic state graphs is presented in Section 3. In Section 4 we show the translation of TPPAL into probabilistic dynamic state graphs also showing the equivalence between both semantics. In Section 5 we specify the AUY protocol with TPPAL and we generate the equivalent probabilistic dynamic state graph. Finally, in Section 6 the conclusions and the future work are presented.

2 Syntax and Semantics of TPPAL

2.1 Syntax

Let Vis be a finite set of actions, and $Act = Vis \cup \{\tau\}$. The syntax of the language TPPAL (Timed and Probabilistic Process ALgebra) is defined by the following BNF expression:

$$
\begin{array}{ll}
T ::= & P \mid N \\
N ::= & stop \mid a < t_1, t_2 >; T \mid \tau; T \mid wait(t); T \mid N \,\square\, N \mid \\
 & N \parallel_A N \mid N \backslash a \mid X \mid \mu X.N \\
P ::= & \sum_{i \in I} [q_i].N \mid P \parallel_A N \mid N \parallel_A P \mid P \parallel_A P \mid P \backslash a \mid \\
 & X \mid \mu X.P
\end{array}
$$

where $A \subseteq Act$, $X \in Idf$ (set of identifiers), $a \in Vis$, $t, t_1 \in \mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$, $t_2 \in \mathbb{R}_0^+ \cup \{\infty\}$ and $q \in (0, 1]$.

The informal interpretation of the operators follows:

$stop$.- This represents a deadlock, i.e. no action can be executed.

$a < t_1, t_2 >; T$.- The action $'a'$ can be executed, after a delay, in the time interval $[t_1, t_2]$. After t_2 we lose the ability to execute that action. As usual, once a has been executed, the process behaves as T.

$\tau; T$.- The internal action (τ) is executed immediately, and the process behaves as T.

$wait(t); T$.- The process waits for $'t'$ units of time, and then it behaves as $\tau; P$.

$N_1 \,\square\, N_2$.- This is essentially the classical external choice operator.

$\sum_{i \in I} [q_i].N_i$.- This represents a probabilistic choice, where the system selects the process N_i to execute using the probabilities q_i.

$T_1 \parallel_A T_2$.- This represents the parallel execution of T_1 and T_2 synchronizing on the actions in A.

$T \backslash a$.- All instances of the action a in T are hidden, and thus they become urgent (τ).

X .- This may represent any process of the language.

$\mu X.T$.- This is the classical recursion operator.

Terms obtained from this syntax will be denoted by TPPA. However, in order to define the corresponding probabilistic dynamic state graph for any term we will need to restrict our attention to regular terms, for which terms $\mu X.P$ must be guarded (all appearances of X in P must be prefixed), and there is neither a parallel operator, nor a hiding operator in P. These terms will be denoted by RTPPA.

2.2 Operational Semantics

We define an operational semantics for the closed terms of this language, which has three kinds of transition rules.

$N \underset{t}{\longmapsto} N'$.- The process N behaves as N' after t units of time.

$N \xrightarrow{a} T'$.- The process N executes the action a and then behaves as T' (no time elapses).

$P \rightarrow_q N$.- The probabilistic process P behaves as N with a probability q, with $q \in (0, 1]$.

Then, we have a group of rules that capture the behavior of probabilistic terms (see Table 2) and another group defining the behavior of non_deterministic processes. These rules are shown in Table 1.

With rule N1 we capture that for the process stop the only possible evolution is the passage of time. Rules N2a-c capture the semantics of the timed prefix. Concretely, rule N2c states that once the time to execute an action has expired, we lose the ability to execute that action. Rules N3 and N4a-b capture the urgent character of internal actions, and the semantics of the wait operator. Rules N5a-b give the classical semantics for the external choice, while N5c states that the time can pass in an external choice only when both components are able to do that. Rules N6a-c are the classical

N1)	stop $\underset{t}{\longmapsto}$ stop $\quad \forall t \in \mathbb{R}_0^+$
N2a)	$a < 0, t >; T \xrightarrow{a} T \quad \forall t \in \mathbb{R}_0^+$
N2b)	$a < t_1, t_2 >; T \underset{t'}{\longmapsto} a < t_1 \dot{-} t', t_2 - t' >; T$ $0 \le t' \le t_2,$ where $x \dot{-} y = \max\{0, x - y\}$
N2c)	$a < 0, 0 >; T \underset{t'}{\longmapsto}$ stop $\quad \forall t' > 0$
N3)	$\tau; T \xrightarrow{\tau} T$
N4a)	$\text{wait}(t); T \underset{t'}{\longmapsto} \text{wait}(t - t'); T, \ 0 \le t' \le t$
N4b)	$\text{wait}(0); T \xrightarrow{\tau} T$
N5a)	$\dfrac{N_1 \xrightarrow{e} T_1}{N_1 \Box N_2 \xrightarrow{e} T_1}$
N5b)	$\dfrac{N_2 \xrightarrow{e} T_2}{N_1 \Box N_2 \xrightarrow{e} T_2}$
N5c)	$\dfrac{N_1 \underset{t'}{\longmapsto} N_1' \quad N_2 \underset{t'}{\longmapsto} N_2'}{N_1 \Box N_2 \underset{t'}{\longmapsto} N_1' \Box N_2'}$
N6a)	$\dfrac{N_1 \xrightarrow{e} T_1 \quad e \notin A}{N_1 \parallel_A N_2 \xrightarrow{e} T_1 \parallel_A N_2}$
N6b)	$\dfrac{N_2 \xrightarrow{e} T_2 \quad e \notin A}{N_1 \parallel_A N_2 \xrightarrow{e} N_1 \parallel_A T_2}$
N6c)	$\dfrac{N_1 \xrightarrow{a} T_1, \quad N_2 \xrightarrow{a} T_2 \quad a \in A}{N_1 \parallel_A N_2 \xrightarrow{a} T_1 \parallel_A T_2}$
N6d)	$\dfrac{N_1 \underset{t'}{\longmapsto} N_1', \quad N_2 \underset{t'}{\longmapsto} N_2'}{N_1 \parallel_A N_2 \underset{t'}{\longmapsto} N_1' \parallel_A N_2'}$
N7a)	$\dfrac{N \xrightarrow{e} T' \quad e \in Act \quad e \neq a}{N \backslash a \xrightarrow{e} T' \backslash a}$
N7b)	$\dfrac{N \xrightarrow{a} T}{N \backslash a \xrightarrow{\tau} T \backslash a}$
N7c)	$\dfrac{N \underset{t}{\longmapsto} N', \neg N \xrightarrow{a} \wedge \neg Q}{N \backslash a \underset{t}{\longmapsto} N' \backslash a}$ where $Q = [\exists\, t' < t, N \underset{t'}{\longmapsto} N'', N'' \xrightarrow{a}]$
N8a)	$\dfrac{N\{\mu X.N/X\} \xrightarrow{e} T'}{\mu X.N \xrightarrow{e} T'}$
N8b)	$\dfrac{N\{\mu X.N/X\} \underset{t}{\longmapsto} N'}{\mu X.N \underset{t}{\longmapsto} N'}$
N9)	$\dfrac{N_1 \underset{t_1}{\longmapsto} N_2 \quad N_2 \underset{t_2}{\longmapsto} N_3}{N_1 \underset{t_1+t_2}{\longmapsto} N_3}$

Table 1. Non-Deterministic Rules

ones for the parallel operator, while N6d is similar to N5c. Semantics of the hiding operator is captured by rules N7a-c, the latter one capturing the fact that once an action has been hidden, it becomes urgent. Notice that rule N7c introduces negative premises; however, by using the same procedure as in [Gro93] consistency of the transition system can be proved. Finally, rules N8a-b capture the semantics for the recursion operator.

Rule P1 gives the classical semantics for the probabilistic choice. The behavior of the parallel operator

P1)	$\dfrac{}{\sum_{i \in I} [q_i] N_i \longrightarrow_{q_j} N_j} \quad \forall j \in I$
P2a)	$\dfrac{P_1 \longrightarrow_q N_1}{P_1 \parallel_A N_2 \longrightarrow_q N_1 \parallel_A N_2}$
P2b)	$\dfrac{P_2 \longrightarrow_q N_2}{N_1 \parallel_A P_2 \longrightarrow_q N_1 \parallel_A N_2}$
P2c)	$\dfrac{P_1 \longrightarrow_{q_1} N_1 \quad P_2 \longrightarrow_{q_2} N_2}{P_1 \parallel_A P_2 \longrightarrow_{q_1 \times q_2} N_1 \parallel_A N_2}$
P3)	$\dfrac{P \longrightarrow_q N}{P \backslash a \longrightarrow_q N \backslash a}$
P4)	$\dfrac{P\{\mu x.P/x\} \longrightarrow_q N}{\mu x.P \longrightarrow_q N}$

Table 2. Probabilistic Rules

with probabilistic arguments is captured by the set of rules $P2$. Rule $P3$ states the behavior of a probabilistic process where an action is hidden. Finally, $P4$ is the classical rule for the recursion.

Let us observe that these rules establish that probabilistic choices are resolved before external choices.

Definition 1 *(Timed and probabilistic computations) Let $T,\ T',\ T'' \in TPPAL$, $w \in (Act \cup \mathbb{R}_0^+ \cup \{[q] \mid q \in (0,1]\})^*$, $e \in Act$, $t \in \mathbb{R}_0^+$ and $q \in (0,1]$.*

1. *Let $T \xrightarrow{e} T'$ be a run of T. Then $T \overset{<e>}{\Longrightarrow}_{0,1} T'$ is a timed and probabilistic computation.*
2. *Let $T \underset{t}{\longmapsto} T'$ be a run of T. Then $T \overset{<t>}{\Longrightarrow}_{t,1} T'$ is a timed and probabilistic computation.*
3. *Let $T \longrightarrow_q N'$ be a run of T. Then $T \overset{<[q]>}{\Longrightarrow}_{0,q} N'$ is a timed and probabilistic computation.*
4. *Let $T \overset{<w>}{\Longrightarrow}_{t,p} T'$ be a timed and probabilistic computation and $T' \xrightarrow{e} T''$ a run of T'. Then $T \overset{<w.e>}{\Longrightarrow}_{t,p} T''$ is a timed and probabilistic computation.*
5. *Let $T \overset{<w>}{\Longrightarrow}_{t,p} T'$ be a timed and probabilistic computation and $T' \underset{t'}{\longmapsto} T''$ a run of T'. Then $T \overset{<w.t'>}{\Longrightarrow}_{t+t',p} T''$ is a timed and probabilistic computation.*
6. *Let $T \overset{<w>}{\Longrightarrow}_{t,p} T'$ be a timed and probabilistic computation and $T' \longrightarrow_q T''$ a run of T'. Then $T \overset{<w.[q]>}{\Longrightarrow}_{t,p \times q} T''$ is a timed and probabilistic computation.*

3 Probabilistic Dynamic State Graphs

Probabilistic dynamic state graphs are essentially automata extended with probabilities and timers, which capture the local time information of each process. These graphs have two kinds of nodes (probabilistic and non-deterministic) and two kinds of edges (probabilistic and non-deterministic). Probabilistic dynamic state graphs here defined include a global clock which captures the time evolution for the whole system, and local timers synchronize with the global clock when necessary, thus maintaining the global time information.

Then, nodes of these graphs are labeled by terms, and non-deterministic edges are labeled by tuples (a, r, S) where a is an action, r is a set of time intervals specifying restrictions for some clocks, and S is a set of timers that must synchronize with the global clock once this transition has been executed. Finally probabilistic edges are labeled by real numbers in the interval $(0, 1]$.

Definition 2 *(Probabilistic dynamic state graphs)* *A probabilistic dynamic state graph is a tuple* $(NT, NP, ET, EP, \lambda, \gamma, n_0, clocks)$*, where:*

- NT *is the set of timed nodes*
- NP *is the set of probabilistic nodes*
- $ET \subseteq NT \times (NT \cup NP)$ *is the set of timed edges*
- $EP \subseteq NP \times (NT \cup NP)$ *is the set of probabilistic edges*
- $\lambda : E \longrightarrow Act_\tau \times Times \times \mathcal{P}(clocks)$ *is a labeling function for the timed edges, where* $Times : clocks \hookrightarrow \mathbb{R}_0^+ \times \mathbb{R}_0^+ \cup \{\infty\}$ *is a partial function that associates a time interval to some of the clocks.*
- $\gamma : EP \longrightarrow (0, 1]$ *is the labeling function for probabilistic edges, where*

$$\forall\, n \in NP : \sum_{e \in EP, n \xrightarrow{e}} \gamma(e) = 1$$

When we represent graphically a probabilistic dynamic state graph we draw a cross inside the probabilistic nodes. Probabilistic edges are labeled by probabilities.

3.1 Semantics

Definition 3 *(States)* *Given a probabilistic dynamic state graph* $G = (NT, NP, ET, EP, \lambda, \gamma, n_0, clocks)$*, we define a state of it by a tuple* (n, C, t) *where* $n \in N$*,* $C : Clocks \longrightarrow \mathbb{R}_0^+$ *(clock values) and* $t \in \mathbb{R}_0^+$ *is the current instant,* $t \geq C(R_g)$*.*

We say that a state s *is a probabilistically stable state if and only if* $n \in NT$*. We use* $s \downarrow$ *to denote that* s *is probabilistically stable state and* $s \uparrow$ *to denote that* s *is not probabilistically stable.*

Given a state $s = (n, C, t)$ and an edge e leaving the node n with label: $\lambda(e) = (a, r, S)$, we define:

$$\begin{aligned} Min(n, C, e) = \;& \max\{(C(R_j) + \alpha_j) \;\dot{-}\; C(R_g) \mid \\ & R_j \in Clocks,\; r(R_j) = [\alpha_j, \beta_j]\} \end{aligned}$$

which represents the minimum delay until this edge can be executed, with respect to the instant at which we reached the node n.

$$\begin{aligned} Max(n, C, e) = \;& \min\{(C(R_j) + \beta_j) - C(R_g) \mid \\ & R_j \in Clocks,\; r(R_j) = [\alpha_j, \beta_j]\} \end{aligned}$$

which represent the maximum delay to execute this edge with respect to the instant at which we reached the node n. If $Max(n, C, e) < 0$ the time to execute that action has already expired, and it cannot therefore be executed. We also define:

$$Min_\tau(n, C) = \begin{cases} \infty \quad \text{if } \nexists\, e \text{ s.t. } n \xrightarrow{e},\; \lambda_1(e) = \tau \\ \min\{Min(n, C, e) | n \xrightarrow{e}, \lambda_1(e) = \tau\} \\ \qquad\qquad \text{otherwise} \end{cases}$$

which represents the maximum delay that we have to execute the visible actions, due to the restrictions imposed by the urgent ones.

According to these definitions, it follows that the action a can be executed at the state $s = (n, C, t)$ if and only if we have an edge e with label $\lambda(e) = (a, r, S)$ leaving n such that:

$$Min(n, C, e) \leq \Delta t \leq Max(n, C, e) \wedge \Delta t \leq Min_\tau(n, C)$$

where $\Delta t = t - C(R_g)$. We will denote this predicate by $act(s, e)$.

Consequently, we may establish the following rules to capture the graph semantics:

$$\textbf{G1)} \quad \frac{s \downarrow, act(s, e)}{s \xrightarrow{e} (m, adjust(C, e, t), t)}$$

$$\textbf{G2)} \quad \frac{s \downarrow,\; t' \geq t\;,\; t' - C(R_g) \leq Min_\tau(n, C)}{s \underset{t'-t}{\longmapsto} (n, C, t')}$$

$$\textbf{G3)} \quad \frac{s \uparrow,\; n \longrightarrow_q n'}{s \longrightarrow_q (n', C, t)}$$

with $s = (n, C, t)$, $e = (n, m) \in E$ and

$$adjust(C, e, t)(R_i) = \begin{cases} t & \text{if } R_i \in \lambda_3(e) \\ C(R_i) & \text{otherwise} \end{cases}$$

Definition 4 *(Timed and probabilistic computations)* *Let* $G = (NT, NP, ET, EP, \lambda, \gamma, n_0, clocks)$*,* $w \in (Act \cup \mathbb{R}_0^+ \cup \{[q] \,|\, q \in (0, 1]\})^*$*,* $e \in Act$*,* $t \in \mathbb{R}_0^+$ *and* $q \in (0, 1]$.

1. *Let $s_0 = (n_0, C_0, t_0)$ be the initial state of G, and $s_0 \xrightarrow{e} s'$ a possible run of G. Then $s_0 \overset{<e>}{\Longrightarrow}_{0,1} s'$ is a timed and probabilistic computation.*

2. *Let $s_0 = (n_0, C_0, t_0)$ be the initial state of G and $s_0 \underset{t}{\longmapsto} s'$ a possible run of G. Then $s_0 \overset{<t>}{\Longrightarrow}_{t,1} s'$ is a timed and probabilistic computation.*

3. *Let $s_0 = (n_0, C_0, t_0)$ be the initial state of G and $s_0 \longrightarrow_q s'$ a possible run of G. Then $s_0 \overset{<[q]>}{\Longrightarrow}_{0,q} s'$ is a timed and probabilistic computation.*

4. *Let $s_0 \overset{<w>}{\Longrightarrow}_{t,p} s'$ be a timed and probabilistic computation of G and $s' \xrightarrow{e} s''$ a possible run of G. Then $s_0 \overset{<w.e>}{\Longrightarrow}_{t,p} s''$ is a timed and probabilistic computation.*

5. *Let $s_0 \overset{<w>}{\Longrightarrow}_{t,p} s'$ be a timed and probabilistic computation of G and $s' \underset{t'}{\longmapsto} s''$ a possible run of G. Then $s_0 \overset{<w.t>}{\Longrightarrow}_{t+t',p} s''$ is a timed and probabilistic computation.*

6. *Let $s_0 \overset{<w>}{\Longrightarrow}_{t,p} s'$ be a timed and probabilistic computation of G and $s' \longrightarrow_q s''$ a possible run of G. Then $s_0 \overset{<w.[q]>}{\Longrightarrow}_{t,p\times q} s''$ is a timed and probabilistic computation.*

4 Process Translation

Now we present the translation of each RTPPA process to its corresponding graph, according to the syntactical structure of the term.

- Stop:

  $$G[\![stop]\!] = (\{stop\}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, stop, \{R_g\})$$

  where, R_g is the global clock.

- Timed prefix
  Let $T = a < t_1, t_2 >; T_1$ be a process with $G[\![T_1]\!] = (NT_1, NP_1, ET_1, EP_1, \lambda_1, \gamma_1, n_{01}, clocks_1)$.

  Then, $G[\![T]\!] = (NT,\ NP_1,\ ET,\ EP_1,\ \lambda,\ \gamma_1,\ P,\ clocks_1)$, where

  - $NT = NT_1 \cup \{P\}$
  - $ET = ET_1 \cup \{P \xrightarrow{e} n_{01}\}$
  - $\lambda(\bar{e}) = \begin{cases} \lambda_1(\bar{e}) & \forall \bar{e} \in ET_1 \\ (a, Intg, clocks_1) & \text{if } \bar{e} = e \end{cases}$

  The function $Intg$ is only defined for R_g, being $Int(R_g) = [t_1, t_2]$.

- Urgent action prefix and wait operators
  The construction of the graph for these operators is similar to the construction for the timed prefix, although the label for the new edge is now τ in both cases . Furthermore, for the urgent action prefix $Int(R_g) = [0, 0]$ and for the wait operator $Int(R_g) = [t, t]$.

- External choice
  $G[\![N_1 \Box N_2]\!] = G[\![Unfold(N_1)]\!] \ \Box_G G[\![Unfold(N_2)]\!]$, where $\Box_G$ is a graph operation defined below that requires both components to have an initial state without incoming edges. Thus, in order to define the general behavior of the external choice we need to *Unfold* the outer recursion that could appear on the external choice components by applying the function *Unfold*, which is defined as follows:

  $Unfold(stop) = stop$
  $Unfold(a < t_1, t_2 >; P) = a < t_1, t_2 >; P$
  $Unfold(\tau; P) = \tau; P$
  $Unfold(wait(t); P) = wait(t); P$
  $Unfold(P \backslash a) = Unfold(P) \backslash a$
  $Unfold(\mu X.P) = Unfold(P\{\mu X.P/X\})$
  $Unfold(P_1 \Box P_2) = Unfold(P_1) \Box Unfold(P_2)$
  $Unfold(P_1 \parallel_A P_2) = Unfold(P_1) \parallel_A Unfold(P_2)$
  $Unfold(\sum_{i \in I} [q_i] N_i) = \sum_{i \in I} [q_i] N_i$

  Note that this definition is correct, because we are considering regular terms.

  Then, given two graphs, $G_i = (NT_i,\ NP_i,\ ET_i,\ EP_i,\ \lambda_i,\ \gamma_i,\ n_{0i},\ clocks_i)$, for $i = 1, 2$, such that their initial states do not have incoming edges, we define $G_1 \Box_G G_2 = (NT,\ NP,\ ET, EP,\ \lambda,\ \gamma,\ n_0,\ Clocks_1 \cup Clocks_2)$ as follows:

  - $NT = (NT_1 \cup NT_2 \cup \{N\}) \setminus \{n_{01}, n_{02}\}$
  - $ET = \begin{array}{l} ((ET_1 \cup ET_2) \setminus \\ (\{(n_{01}, n_1) \mid (n_{01}, n_1) \in ET_1\} \cup \\ \{(n_{02}, n_2) \mid (n_{02}, n_2) \in ET_2\})) \cup \\ (\{(n_0, n_1) \mid (n_{01}, n_1) \in ET_1\} \cup \\ \{(n_0, n_2) \mid (n_{02}, n_2) \in ET_2\}) \end{array}$
  - $\lambda(e) = (\bar{\lambda_1}(e), \bar{\lambda_2}(e), \bar{\lambda_3}(e) \cup \{R_g\}) \ \ \forall e \in ET$
    where R_g is either the global clock of G_1 or the global clock of G_2, which becomes the global clock of G, and $\bar{\lambda}_j$, for $j = 1, 2, 3$, is defined by:
  - $\bar{\lambda_j}(e) = \begin{cases} \lambda_{ij}(e) & \forall e \in ET_i \cap ET \\ \lambda_{ij}(e_i) \text{ if } e = (n_0, n_i) \wedge e_i = (n_{0i}, n_i) \end{cases}$
    where λ_{ij} stands for the j-component of λ_i.
  - $NP = NP_1 \cup NP_2$, $EP = EP_1 \cup EP_2$ and
    $\gamma(e) = \begin{cases} \gamma_1(e) & \text{if } e \in EP_1 \\ \gamma_2(e) & \text{if } e \in EP_2 \end{cases}$

- Probabilistic choice
  In this case we do not need to impose the restriction which forbids the initial nodes of the graphs taking part in the choice to have incoming edges. This is because when we construct the graph we do not change the destination node of none of the edges of the involved graphs; instead we add some new edges. These new edges link the new node with the initial nodes of the involved graphs. Therefore, it is not necessary to apply the function *Unfold* to the subterms of the choice.

  Let $G[\![N_i]\!] = (NT_i, NP_i, ET_i, EP_i, \lambda_i, \gamma_i, n_{0i}, \mathit{clocks}_i)$ be a graph for a process N_i The graph of $\sum_{i \in I} [q_i] N_i$ is constructed by adding a new probabilistic node and many probabilistic edges as processes take part in the choice. These edges will start from the new node and end in the initial node n_{0i} of each graph, and they will be labeled with the probability q_i associated to the process.

  Then, $G[\![\sum_{i\in I} [q_i]N_i]\!] = (\bigcup_{i\in I} NT_i, \bigcup_{i\in I} NP_i \cup \{P\}, \bigcup_{i\in I} ET_i, EP, \lambda, \gamma, n_0, \bigcup_{i\in I} \mathit{clocks}_i)$, where $EP = \bigcup_{i\in I} EP_i \bigcup_{i\in I} (P \longrightarrow_{q_i} n_{0i})$,
  $\lambda(e) = (\lambda_1(e), \lambda_2(e), \lambda_3(e) \cup \{R_g\}) \quad \forall\, e \in ET$, and

  $$\gamma(e) = \begin{cases} \gamma_i(e) & \text{if } e \in EP_i \\ q_i & \text{if } e \in EP \setminus EP_i \end{cases}$$

  Just like for the external choice, the global clock in the new graph can be whichever of the involved graphs $G[\![N_i]\!]$.

- Parallel Composition
  Let $T = T_1 \|_{\{A\}} T_2$, with $G[\![T_i]\!] = (NT_i, NP_i, ET_i, EP_i, \lambda_i, \gamma_i, n_{0i}, \mathit{clocks}_i)$, where $i = 1, 2$. Then

  $$G[\![T]\!] = (NT, NP, ET, EP, \lambda, \gamma, n_0, \mathit{clocks})$$

  This graph is obtained by applying the Cartesian product of $G[\![T_1]\!]$ and $G[\![T_2]\!]$. In the new graph:

  - The set of clocks *clocks* consists of the clock of both graphs and a new one that is the global clock of the whole graph. This one is added in the label of each timed edge of the graph.
  - The set of timed nodes will consist of all the nodes obtained from the Cartesian product of the timed nodes. The set of probabilistic nodes will consist of the nodes obtained either from a timed node and a probabilistic one, or from two probabilistic ones. The initial node of the new graph will be the one obtained from the initial nodes of both graphs. This node can be therefore timed or probabilistic.
    * $NT = NT_1 \times NT_2$
    * $NP = (NT_1 \times NP_2) \cup (NP_1 \times NT_2) \cup (NP_1 \times NP_2)$
    * $n_0 = (n_{01}, n_{02})$
  - The set of timed edges will consist of the edges of both graphs whose initial node in the new graph is a timed one. The edges of both graphs labeled with the same synchronization action will be merged into an only edge. The timed edges whose initial node in the new graph is a probabilistic one are removed. Then, the definition for the set of timed edges is the following:

    $$ET = ET_{11} \cup ET_{12} \cup ET_{13}$$

    where:

    $$\begin{aligned} ET_{11} = \ & \{((n_1, n_2), (n_1', n_2)) \mid (n_1, n_2) \in NT, \\ & n_1 \xrightarrow{e_1}_1 n_1',\ \lambda_{11}(e_1) \notin A\} \\ ET_{12} = \ & \{((n_1, n_2), (n_1, n_2')) \mid (n_1, n_2) \in NT, \\ & n_2 \xrightarrow{e_2}_2 n_2',\ \lambda_{21}(e_2) \notin A\} \\ ET_{13} = \ & \{((n_1, n_2), (n_1', n_2')) \mid (n_1, n_2) \in NT, \\ & n_1 \xrightarrow{e_1}_1 n_1',\ n_2 \xrightarrow{e_2}_2 n_2', \\ & \lambda_{11}(e_1) = \lambda_{21}(e_2) \in A\} \end{aligned}$$

  - $EP = EP_{11} \cup EP_{12} \cup EP_{13}$ where:

    $$\begin{aligned} EP_{11} = \ & \{((n_1, n_2), (n_1', n_2)) \mid n_2 \in NT_2, \\ & (n_1, n_1') \in EP_1\} \\ EP_{12} = \ & \{((n_1, n_2), (n_1, n_2')) \mid n_1 \in NT_1, \\ & (n_2, n_2') \in EP_2\} \\ EP_{13} = \ & \{((n_1, n_2), (n_1', n_2')) \mid \\ & (n_1, n_1') \in EP_1,\ (n_2, n_2') \in EP_2\} \end{aligned}$$

  - The labeling functions for the edges will be the following:

    $$\lambda(e) = \begin{cases} (\lambda_{11}(e_1), \lambda_{12}(e_1), \lambda_{13}(e_1) \cup \{R_g\}) \\ \qquad \text{for } e_1 \in ET_1,\ e \in ET_{11} \\ (\lambda_{21}(e_2), \lambda_{22}(e_2), \lambda_{23}(e_2) \cup \{R_g\}) \\ \qquad \text{for } e_2 \in ET_2,\ e \in ET_{12} \\ (a, \lambda_{12}(e_1) \cup \lambda_{22}(e_2), \lambda_{13}(e_1) \cup \\ \lambda_{23}(e_2) \cup \{R_g\}) \\ \qquad \text{for } e \in ET_{13}, a = \lambda_{11}(e_1) \end{cases}$$

    and

    $$\gamma(e) = \begin{cases} \gamma_1(e_1) & \text{if } e_1 \in EP_1,\ e \in EP_{11} \\ \gamma_2(e_2) & \text{if } e_2 \in EP_2,\ e \in EP_{12} \\ \gamma_1(e_1) \times \gamma_2(e_2) & \text{if } e \in EP_{13} \end{cases}$$

- Hiding
  Let $T = T_1 \backslash a$, be a process with $G[\![T_1]\!] = (NT_1, NP_1, ET_1, EP_1, \lambda_1, \gamma_1, n_{01}, \mathit{clocks}_1)$.

  Then $G[\![T]\!] = (NT_1 \backslash a, NP_1 \backslash a, ET_1, EP_1, \lambda, \gamma_1, n_{01}, \mathit{clocks}_1)$, where

$$\lambda(e)=\begin{cases} (\tau, \lambda_{12}(e), \lambda_{13}(e)) & \forall e \in ET_1, \lambda_{11}(e) = a \\ \lambda_1(e) & \text{otherwise} \end{cases}$$

$NT_1\backslash a$ and $NP_1\backslash a$ represent the terms of NT_1 and NP_1, where the subterm $\backslash a$ has been added.

- X process
  $G[\![X]\!] = (\{X\}, \emptyset, \emptyset, \emptyset, \emptyset, X, \{R_g\})$, where R_g is the global clock.

- Recursion
  Let $G[\![T_1]\!] = (NT_1, NP_1, ET_1, EP_1, \lambda_1, \gamma_1, n_{01}, \mathit{clocks}_1)$ be the graph associated to T_1.

  Then $G[\![\mu X.T_1]\!] = (NT, NP_1, ET, EP, \lambda, \gamma, n_{01}, \mathit{clocks}_1)$, where

  – the set of timed nodes will be the set of timed nodes in $G[\![T_1]\!]$, where the nodes labeled with X have been removed, and each occurrence of X in the terms labeling the nodes is changed by $\mu X.T_1$.
  $NT = NT_1 \setminus \{n \in NT_1 \mid \mathit{term}(n) = X\}$
  In the set of edges we just change the final node of the edges whose final node was removed, their new final one being the initial node of $G[\![T_1]\!]$.

  $$\begin{aligned} ET = \quad & ET_1 \cup \{(n, n_{01}) \mid (n,m) \in ET_1 \wedge \\ & \mathit{term}(m) = X\} \backslash \\ & \{(n,m) \in ET \mid \mathit{term}(m) = X\} \\ EP = \quad & EP_1 \cup \{(n, n_{01}) \mid (n,m) \in EP_1 \wedge \\ & \mathit{term}(m) = X\} \backslash \\ & \{(n,m) \in EP_1 \mid \mathit{term}(m) = X\} \end{aligned}$$

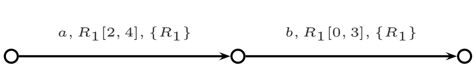
  For the edges whose final node has been changed we must redefine the labeling function, replacing the set of clocks that synchronize with the global one by clocks_1 Then, the new labeling function is defined by:

  $$\lambda(e) = \begin{cases} \lambda_1(e) & \text{if } e \in ET_1 \\ (\lambda_{11}(e), \lambda_{21}(e), \mathit{clocks}_1) & \\ \quad \text{if } e = (n, n_{01}) \wedge (n,m) \in ET_1, & \\ \quad \mathit{term}(m) = X & \end{cases}$$

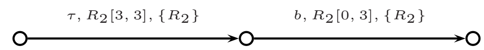
4.1 Examples

In this section we present some RTPPA processes and their corresponding probabilistic dynamic state graphs according to the previous definition:

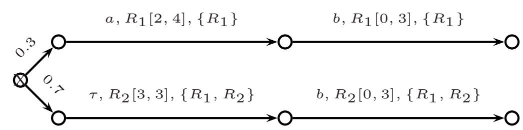
- $P_1 = a < 2, 4 >; b < 0, 3 >; \mathit{stop}$



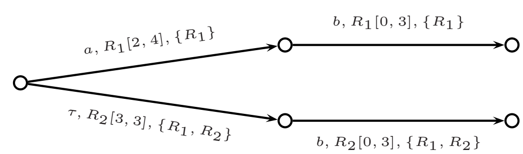
- $P_2 = \mathit{wait}(3); b < 0, 3 >; \mathit{stop}$



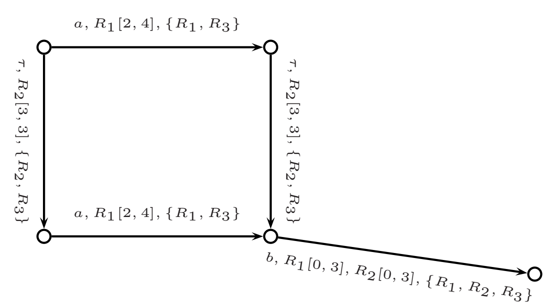
- $P_3 = [0, 3]P_1 + [0, 7]P_2$



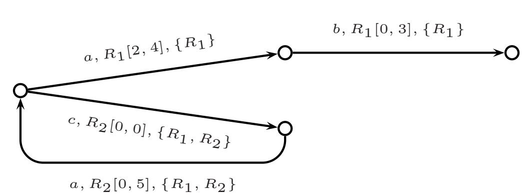
- $P_4 = P_1 \Box P_2$



- $P_5 = P_1 \|_{\{b\}} P_2$



- $P_6 = \mu X.(P_1 \Box c < 0, 0 >; a < 0, 5 >; X)$



4.2 Equivalence

In order to establish the correspondence between the terms that we can reach according to the operational semantics and the reachable states in the probabilistic dynamic state graph associated to the given process, we first need to capture the reachable states from a given node by time elapsing. However, we must take into account that every sequential component of a process has its own clock, and they are updated separately. Consequently, the evolution in time from a node must be defined by distinguishing the separate evolutions in time of each sequential component.

Definition 5 *(Sequential components) We define the set of sequential components as follows:*

$$\begin{array}{rl} S ::= & stop | a < t_1, t_2 >; T | \tau; T | wait(t); T | S \Box S | \\ & \sum_{i \in I} [q_i] S_i | S \|_A | \ _A\| S \ | \ S \backslash a \end{array}$$

with $TP \in TPPA$*. We then define the set of sequential components of a closed regular process* P *as follows:*

$$\begin{array}{l} dex(stop) = \{stop\} \\ dex(a < t_1, t_2 >; T) = \{a < t_1, t_2 >; T\} \\ dex(wait(t); T) = \{wait(t); T\} \\ dex(\tau; T) = \{\tau; T\} \\ dex(N_1 \Box N_2) = dex(N_1) \Box dex(N_2) \\ dex(\sum_{i \in I} [q_i] N_i) = \sum_{i \in I} dex[q_i](N_i) \\ dex(P_1 \|_A P_2) = dex(P_1) \|_A \cup \ _A\| dex(P_2) \\ dex(P \backslash a) = dex(P) \backslash a \\ dex(\mu X.P) = dex(P\{\mu X.P/X\}) \end{array}$$

Once we have defined the decomposition in sequential processes, we can define the function *"time_evol"*, which captures the possible terms that we may reach by time elapsing on each sequential component of a process, as follows:

Definition 6 *Given two closed regular processes,* P, Q*, we say that* $P \in time\_evol(Q)$ *if and only if there is bijection* $\varphi : dex(Q) \longrightarrow dex(P)$*, s.t.* $\forall S_Q \in dex(Q)\ \exists r \in \mathbb{R}_0^+$*,* $old(S_Q, r) = \varphi(S_Q)$*, where old is defined as follows:*

$$old(stop, r) = stop$$

$$old(a < t_1, t_2 >; P, r) = \begin{cases} stop & \text{if } r > t_2 \\ a < t_1 \dot{-} r, t_2 - r >; P & \\ \quad \text{if } r \leq t_2 & \end{cases}$$

$$old(wait(t); P, r) = \begin{cases} stop & \text{if } r > t \\ wait(t - r); P & \text{if } r \leq t \end{cases}$$

$$old(\tau; P, r) = \begin{cases} stop & \text{if } r > 0 \\ \tau; P & \text{if } r = 0 \end{cases}$$

$$old(S_1 \Box S_2, r) = old(S_1, r) \Box old(S_2, r)$$

$$old(\sum_{i \in I} [q_i] N_i, r) = \begin{cases} stop & \text{if } r > 0 \\ \sum_{i \in I} [q_i] N_i & \text{if } r = 0 \end{cases}$$

$$\begin{array}{l} old(S \|_A, r) = old(S, r) \|_A \\ old(\ _A\| S, r) = \ _A\| old(S, r) \\ old(S \backslash a, r) = old(S, r) \backslash a \end{array}$$

We may now state our two main results, which establish the equivalence between the operational semantics and the graph semantics.

Theorem 1 *Given two closed regular processes* $T, T' \in RTPPA$*, if we have* $T \stackrel{<s>}{\Longrightarrow}_{t,p} T'$*, then there is a term* $\bar{T}' \in TPPA$ *and a node* $n_{\bar{T}'} \in G[\![T]\!]$ *such that* $T' \in time\_evol(\bar{T}')$*, and there is a state* $(n_{\bar{T}'}, C, t)$ *in* $G[\![T]\!]$ *such that*

$$(n_0, C_0, 0) \stackrel{<s>}{\Longrightarrow}_{t,p} (n_{\bar{T}'}, C, t)$$

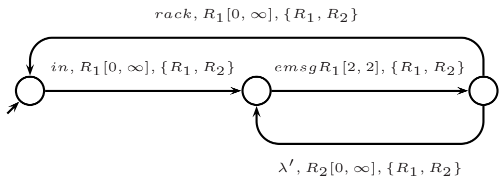
.

Note that the converse is not true in general, because not every $P' \in time\_evol(\bar{P}')$ can be reached by using the operational semantics. Instead, we have the following result for the converse.

Theorem 2 *Given a closed regular process* $T \in RTPPA$*, if* $(n_0, C_0, 0) \stackrel{<s>}{\Longrightarrow}_{t,p} (n_{\bar{T}'}, C, t)$ *then there is a regular process* $T' \in RTPPA$ *such that* $T \stackrel{<s>}{\Longrightarrow}_{t,p} T'$*, where* $T' \in time\_evol(\bar{T}')$*, and* s *is the sequence of transitions executed in* $G[\![P]\!]$ *to reach that state.*

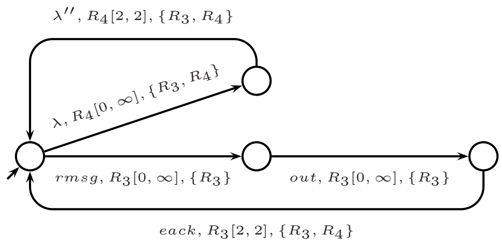
5 Example: AUY Protocol

In this section we use the TPPAL language to model the AUY-Protocol, proposed by Aho, Ullman and Yannakakis in [AUY79], which ensures a reliable communication in a system where channels may fail. The system consists of both a sender and a receiver communicating messages each other through two unreliable channels, which can lose messages with a known probability $p = 0.2$. We assume that the transmission delay for both channels is 2 units of time, and we suppose that when the first channel fails, it replaces the message by a special message λ (after a time-out of 4 units of time), which is sent to the receiver. Once the receiver gets a message, if it is not a λ, it sends an acknowledgment message to the sender through the second channel, which can lose it again with probability p. In such a case, this channel replaces the *ack* message by a λ' message, which is sent to the sender after 4 units of time. On the other hand, if the receiver gets a λ message, it sends a bad message λ'' through the second channel, which causes this channel to generate a λ' message for the sender (after 3 units of time).



$clocks = \{R_1, R_2\}$ $R_g = R_1$

Figure 1. Graph for the sender



$clocks = \{R_3, R_4\}$ $R_g = R_3$

Figure 2. Graph for the receiver

Then, once the sender gets a λ' message, it resends the original message through the first channel.

The specification for this protocol is the following:

- Sender

$$\begin{array}{ll} T = & in < 0,\infty >; T_1 \\ T_1 = & emsg < 2,2 >; (rack < 0,\infty >; T \Box \\ & \lambda' < 0,\infty >; T_1) \end{array}$$

- Channel

$$\begin{array}{ll} C_1 = & emsg < 0,\infty >; ([0,2]\lambda < 4,4 >; C_1 + \\ & [0,8]rmsg < 0,0 >; C_1 \\ C_2 = & (eack < 0,\infty >; ([0,2]\lambda' < 4,4 >; C_2 + \\ & [0,8]rack < 0,0 >; C_2))\Box \\ & (\lambda'' < 0,\infty >; \lambda' < 3,3 >; C_2) \end{array}$$

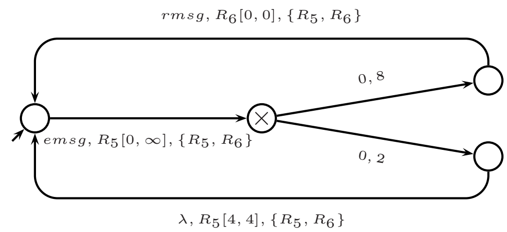
- Receiver

$$\begin{array}{l} R = \quad (rmsg < 0,\infty >; out < 0,\infty >; \\ eack < 2,2 >; R)\Box(\lambda < 0,\infty >; \lambda'' < 2,2 >; R) \end{array}$$

- AUY- Protocol

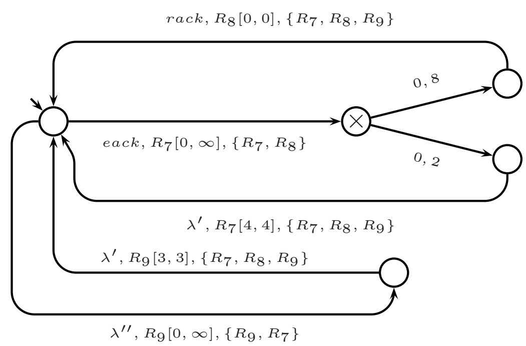
$$\begin{array}{ll} AUY = & (T\|_{\{emsg,rack,\lambda'\}}(C_1\|C_2)) \\ & \|_{\{rmsg,eack,\lambda,\lambda''\}}R \end{array}$$

The corresponding graphs for these processes are shown in Fig. 1,2, 3 and 4.



$clocks = \{R_5, R_6\}$ $R_g = R_5$

Figure 3. Graph for the channel C1



$clocks = \{R_7, R_8, R_9\}$ $R_g = R_9$

Figure 4. Graph for the channel C2

6 Conclusions

In this paper we have defined a Timed and Probabilistic Process Algebra, which is based on basic LOTOS, by adding probabilities and some new operators for the specification of timed behaviors and urgent interactions. We have defined an operational semantics for the language, and a translation into Probabilistic Dynamic State Graphs, which allows us to simulate the specifications in an easy way, as well as to verify some properties of interest. Now we are implementing a tool that supports this language, in order to generate and simulate these graphs. This translation works fine for regular terms, for which we obtain a finite graph. We have proved that both semantics are equivalent, in the sense that every sequence of transitions that we may execute according to the operational semantics can also be executed in the corresponding graph, and conversely.

Our future work will focus on the analysis of properties by using the graphs. Another aspect that we intend to develop is the modular specification of components, in order to reduce the complexity of the obtained graphs.

References

[And00] S. Andova. Time and probability in process algebra. In *8th AMAST, LNCS 1816*, pages 323–338. Springer, 2000.

[AUY79] A. V. Aho, J. D. Ullman, and M. Yannakakis. Modeling communication protocols by automata. In *Proceedings of the 20th FOCS, 1979*, 1979.

[BA03] M. Bravetti and A. Aldini. Discrete time generative-reactive probabilistic processes with different advancing speeds. *Theoretical Computer Science*, 290(1):355–406, 2003.

[BL91] T. Bolognesi and F. Lucidi. Timed process algebras with urgent interactions and a unique powerful binary operator. *Proc. of REX Workshop (Real-Time: Theory and Practice)*, 1991.

[CdFV97] F. Cuartero, D. de Frutos, and V. Valero. A Sound and Complete Proof System for Probabilistic Processes. In *ARTS'96, LNCS 1231*, 1997.

[Gro93] Jan Friso Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118:263–299, 1993.

[Han92] H. Hansson. Modeling timeouts and unreliable media with a timed probabilistic calculus. *Formal Description Techniques IV*, 1992.

[HJ90] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. *Proceedings Real-Time Systems Symposium*, 1990.

[LS89] K.G. Larsen and A. Skou. Bisimulation through Probabilistic Testing. In *Proceedings of 16th ACM Symp. on Principles of Programming Languages*, Austin, TX, 1989.

[MT90] F. Moller and C. Tofts. A temporal calculus of communicating systems. *Proceedings CONCUR 90, Lecture Notes in Computer Science*, (458):401–415, 1990.

[Nuñ03] M. Nuñez. Algebraic Theory of Probabilistic Processes. *The Journal of Logic and Algebraic Programming*, 56(1–2):117–177, May-August 2003.

[Ort90] Yolanda Ortega. *En busca del Tiempo Perdido (In Spanish)*. PhD. Thesis, Univ. Complutense de Madrid, Dpto. Inf. y Automática UCM, 1990.

[QAdF89] J. Quemada, A. Azcorra, and D. de Frutos. A timed calculus for lotos2. *Proc. FORTE 89*, 1989.

[RR88] G. M. Reed and A. W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, (58):249–261, 1988.

[Sch95] S. Schneider. An operational semantics for timed csp. *Information and Computation*, (116):193–213, 1995.