Specifying the Memorization Process with STOPA*

Fernando L. Pelayo
Dept. Informática
E. Politécnica Superior de Albacete
Universidad de Castilla-La Mancha
Campus Universitario. E-02071 Albacete. Spain
fpelayo@info-ab.uclm.es

Manuel Núñez, Natalia López
Dept. Sistemas Informáticos y Programación
Facultad Informática
Universidad Complutense de Madrid
C/. Juan del Rosal, 8. E-28040 Madrid. Spain
{mn,natalia}@sip.ucm.es

Abstract

In this paper we use the formal language STOPA for specifying cognitive systems. In addition to the usual characteristics of these formalisms, our language features the possibility of including stochastic time. This kind of time is useful to represent systems where the delays are not controlled by fix amounts of time, but they are given by a probability distribution function.

In order to illustrate the usefulness of our formalism we will formally represent a cognitive model of the memory. Following contemporary theories of memory classification (see [8, 7]) we consider sensory buffer, short-term, and long-term memories. Moreover, borrowing from [12], we also consider the so-called action buffer memory.

1. Introduction

Cognitive informatics studies intelligent behaviour using computing technology in terms of updated research efforts and progresses of brain science and neural science. We can view it as a branch of cognitive science. From the point of view informatics, we can divide it into external and internal information processing. Previous information processing is focuses on external information processing. Cognitive informatics is going to research on the internal information processing mechanisms and natural intelligence in the brain. It is expected that the investigation into cognitive informatics will result in fundamental findings towards the intelligence essence and new generation of information technology.

Comparing with a computer system, the brain can be taken as hardware and the mind looks like software. The mind contains perception, rational, consciousness and emotion. Cognitive informatics will investigate basic issues, such as learning, memory, thought, language, and neural computing. For more details see [13, 14]

Since Cognitive informatics is a relatively new field of knowledge, new mechanisms are being introduced in the field. The introduction of RTPA [10, 9, 11] represents a very adequate step in this direction. By conveniently putting together the ideas underlying the definition of classical process algebras, RTPA is a new mathematical framework to represent cognitive processes and systems.

For the sake of explaining the reasons for having choose STOPA when specifying the cognitive process of memorization, a reading of [2] should be done. There is justified the choosing of stochastic time.

In this paper we build, on the one hand, on our work on stochastic and probabilistic process algebras (see for example [5, 1, 4, 3]) and, on the other hand, on the process algebra RTPA [10, 11] to introduce a new stochastic process algebra to formally represent cognitive pro-

* Research partially supported by the Spanish MCYT project TIC2003-07848-C02, the Junta de Castilla-La Mancha project PAC-03-001, and the Marie Curie project MRTN-CT-2003-505121/TAROT.

cesses containing stochastic information. We call this language STOPA. The main improvement of our framework with respect to RTPA is that we may specify timed information given by stochastic delays. Nevertheless, let us remark that deterministic delays (as presented in timed process algebras) can be specified by using Dirac distributions. As we will see in the following sections, the inclusion of stochastic time introduces some additional complexity in the definition of the operational semantics of the language.

In order to asses the usefulness of our language, we formally represent a high level description of the cognitive process of memory. We follow contemporary theories of memory classification already stated in [8, 7, 12].

2. The language STOPA

In this section we present our language and its operational semantics. The semantic model is strongly based on [10]. Our modifications are given mainly in the presentation of the operational semantics for process relations and the introduction of the stochastic time.

A process can be defined as a single *meta-process* or as a complex process based on meta-processes using *process relations*. Thus, STOPA is described by using the following structure:

STOPA ::= Meta-processes
| Primary types
| Abstract data types
| Process relations
| System architectures
| Specification refinement sequences

The syntax and operational semantics of the meta-processes are given in Tables 1 and 2. The definition of the primary data types are the meta-types defined from 2.1 to 2.10 in Table 3. The abstract data types are described in Table 4. The interested reader may find in [10] a detailed explanation of all the definitions appearing in Tables 1-4 as weel as the definitions of *system architectures* and *specification refinement sequences*. In Section 3 we will describe the syntax and operational semantics of the process relations.

As we have already commented in the introduction of this paper, our aim is to add stochastic information in the framework of RTPA. For that reason we have included random variables in the syntax of the process relations, that is, a process can be delayed according to a random variable. We will suppose that the sample space (that is, the domain of random variables) is the set of real numbers $\mathbf{R}$ and that random variables take only positive values, that is, given a random variable ξ we have $\mathrm{P}(\xi \leq t) = 0$ for any $t \leq 0$. The reason for this restriction is that random variables are always associated with time distributions.

Definition 1 Let ξ be a random variable. We define its *probability distribution function*, denoted by F_ξ, as the function $F_\xi : \mathbf{R} \longrightarrow (0, 1]$ such that $F_\xi(x) = \mathrm{P}(\xi \leq x)$, where $\mathrm{P}(\xi \leq x)$ is the probability that ξ assumes values less than or equal to x. □

The following probability distribution functions will be used in the rest of the paper.

Uniform Distributions. Let $a, b \in \mathbf{R}^+$ such that $a < b$. A random variable ξ follows a uniform distribution in the interval $[a, b]$, denoted by $U(a, b)$, if its associated probability distribution function is:

$$F_\xi(x) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{x-a}{b-a} & \text{if } a < x < b \\ 1 & \text{if } x \geq b \end{cases}$$

These distributions allow us to keep compatibility with time intervals in timed process algebras in the sense that the same *weight* is assigned to all the times in the interval.

Discrete Distributions. Let $P_I = \{(t_i, p_i)\}_{i \in I}$ be a set of pairs such that for any $i \in I$ we have that $t_i \geq 0$, $p_i > 0$, for any $i, j \in I$, if $i \neq j$ then $t_i \neq t_j$, and $\sum p_i = 1$. A random variable ξ follows a discrete distribution with respect to P_I, denoted by $D(P_I)$, if its associated probability distribution function is:

$$F_\xi(x) = \sum_{i \in I} \{\!| p_i \mid x \geq t_i |\!\}$$

Let us note that $\{\!|$ and $|\!\}$ represent multisets. Discrete distributions are important because they allow us to express *passive* actions, that is, actions that are willing, from a certain point of time, to be executed with probability 1.

Exponential Distributions. Let $0 < \lambda \in \mathbf{R}$. A random variable ξ follows an exponential distribution with parameter λ, denoted by $E(\lambda)$ or simply λ, if its associated probability distribution function is:

$$F_\xi(x) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Poisson Distributions. Let $0 < \lambda \in \mathbf{R}$. A random variable ξ follows a Poisson distribution with parameter λ, denoted by $P(\lambda)$, if it takes positive values only in $\mathbf{N}$ and its associated probability distribution function is:

$$F_\xi(x) = \begin{cases} \sum_{t \in \mathbf{N}, t \leq x} \frac{\lambda^t}{t!} e^{-\lambda t} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

No.	Meta-process	Syntax	Operational Semantics
1.1	System	$\S(SysID_S)$	Represents a system, $SysID$, identified by a string, S
1.2	Assignment	$y_{Type} := x_{Type}$	if $x.type = y.type$ then $x.value \Rightarrow y.value$ else !($@\text{AssigmentTypeError}_S$) where $Type \in \text{Meta} - \text{Types}$
1.3	Addressing	$ptrP\hat{} := x_{Type}$	if $prt.type = x.type$ then $x.value \Rightarrow ptr.value$ else !($@\text{AssigmentTypeError}_S$) where $Type \in \{H, Z, P\hat{}\}$
1.4	Input	$\text{Port}(ptrP\hat{})_{Type}\| > x_{Type}$	if $\text{Port}(ptrP\hat{}).type = x.type$ then $\text{Port}(ptrP\hat{}).value \Rightarrow x.value$ else !($@\text{InputTypeError}_S$) where $Type \in \{B, H\}$, $P\hat{} \in \{H, N, Z\}$
1.5	Output	$x_{Type}\| < \text{Port}(ptrP\hat{})_{Type}$	if $\text{Port}(ptrP\hat{}).type = x.type$ then $x.value \Rightarrow \text{Port}(ptrP\hat{}).value$ else !($@\text{OutputTypeError}_S$) where $Type \in \{B, H\}$, $P\hat{} \in \{H, N, Z\}$
1.6	Read	$\text{Mem}(ptrP\hat{})_{Type} > x_{Type}$	if $\text{Mem}(ptrP\hat{}).type = x.type$ then $\text{Mem}(ptrP\hat{}).value \Rightarrow x.value$ else !($@\text{ReadTypeError}_S$) where $Type \in \{B, H\}$, $P\hat{} \in \{H, N, Z\}$
1.7	Write	$x_{Type} < \text{Mem}(ptrP\hat{})_{Type}$	if $\text{Mem}(ptrP\hat{}).type = x.type$ then $x.value \Rightarrow \text{Mem}(ptrP\hat{}).value$ else !($@\text{WriteTypeError}_S$) where $Type \in \{B, H\}$, $P\hat{} \in \{H, N, Z\}$
1.8	Timing	a) $@t_{\texttt{hh:mm:ss:ms}} := \S t_{\texttt{hh:mm:ss:ms}}$ b) $@t_{\texttt{yy:MM:dd}} := \S t_{\texttt{yy:MM:dd}}$ c) $@t_{\texttt{yy:MM:dd:hh:mm:ss:ms}} := \S t_{\texttt{yy:MM:dd:hh:mm:ss:ms}}$	if $@t.type = \S t.type$ then $\S t.value \Rightarrow @t.value$ else !($@\text{TimingTypeError}_S$) where $\texttt{yy} \in \{0, \ldots, 99\}$, $\texttt{MM} \in \{1, \ldots, 12\}$, $\texttt{dd} \in \{1, \ldots, 31\}$, $\texttt{hh} \in \{0, \ldots, 23\}$, $\texttt{mm}, \texttt{ss} \in \{0, \ldots, 59\}$, $\texttt{ms} \in \{0, \ldots, 999\}$
1.9	Duration	$@tn_Z := \S tn_Z + \Delta n_Z$	if $@tn.type = \Delta n.type = \S tn.type = Z$ then $(\S tn.value + \Delta n.value)$ mod MaxValue $\Rightarrow @tn.value$ where MaxValue is the upper bound of the system relative-clock, and the unit of all values is ms
1.10	Memory allocation	AllocateObject (ObjectID_S, NofElements_N, ElementType_{RT})	$n_N := \text{NofElements}$; $\mathcal{R}_{i=1}^{n}(\text{new ObjectID}(i_N) : \text{ElementType}_{RT})$; Ⓢ $\text{ObjectID.Existed}_{BL} := \texttt{true}$

Table 1: Meta-processes (1/2).

No.	Meta-process	Syntax	Operational Semantics
1.11	Memory release	ReleaseObject($\mathrm{ObjectID}_S$)	delete $\mathrm{ObjectID}_S$// System.Garbage Collection( ) ; $\mathrm{ObjectID}_S :=$ null ; ⓢ $\mathrm{ObjectID.Release}_{BL} :=$ `true`
1.12	Increase	$\uparrow (n_{Type})$	if $n.value <$ MaxValue then $n.value + 1 \Rightarrow n.value$ else !($@\mathrm{ValueOutofRange}_S$) where $Type \in \{N, Z, B, H, P\hat{}\}$ MaxValue = min{run-time defined upper bound, nature upper bound of $Type$}
1.13	Decrease	$\downarrow (n_{Type})$	if $n.value > 0$ then $n.value - 1 \Rightarrow n.value$ else !($@\mathrm{ValueOutofRange}_S$) where $Type \in \{N, Z, B, H, P\hat{}\}$
1.14	Exception detection	$!(@e_S)$	$\uparrow (\mathrm{ExceptionLogPtr}_{P\cdot})$; $@e_S \Rightarrow$ Mem($\mathrm{ExceptionLogPtr}_{P\cdot})_S$
1.15	Skip	$\emptyset$	Exit a current control structure, such as loop, branch, or switch
1.16	Stop	stop	System stop

Table 2: Meta-processes (2/2).

During the rest of the paper, mainly in the examples and when no confusion arises, we will identify random variables with their probability distribution functions.

Regarding *communication* actions, they can be divided into *input* and *output actions*. Next, we define our alphabet of actions.

Definition 2 We consider a set of *communication* actions $Act = \mathtt{Input} \cup \mathtt{Output}$, where we assume $\mathtt{Input} \cap \mathtt{Output} = \emptyset$. We suppose that there exists a bijection $f : \mathtt{Input} \longrightarrow \mathtt{Output}$. For any *input* action $a? \in \mathtt{Input}$, $f(a?)$ is denoted by the *output* action $a! \in \mathtt{Output}$. If there exists a message transmission between $a?$ and $a!$ we say that a is the *channel* of the communication ($a, b, c, \cdots$ to range over Act).

We also consider a special action $\tau \notin Act$ that represents internal behaviour of the processes. We denote by Act_τ the set $Act \cup \{\tau\}$ ($\alpha, \beta, \gamma, \cdots$ to range over Act_τ).

Besides, we consider a denumerable set Id of process identifiers. In addition, we denote by $\mathcal{V}$ the set of random variables ($\xi, \xi', \psi, \cdots$ to range over $\mathcal{V}$). □

3. Process Relations

In this section, we briefly define the syntax and operational semantics of our process algebra to describe process relations. In this part of the description of the language, operational behaviours will be defined by means of transitions $P \xrightarrow{\omega} P'$ that each process can execute. These are obtained in a structured way by applying a set of inference rules [6]. The intuitive meaning of a transition as $P \xrightarrow{\omega} P'$ is that the process P may perform the action ω and, after this action is performed, then it behaves as P'. Let us note that labels appearing in these operational transitions have the following types: $a? \in \mathtt{Input}$, $a! \in \mathtt{Output}$, $a \in Act$, $\alpha \in Act \cup \{\tau\}$, $\omega \in Act_\tau \cup \mathcal{V}$, and $\xi \in \mathcal{V}$.

The set of process relations, as well as their operational semantics, is given in Table 5. A detailed explanation of the intuitive meaning of each operator, as well as the justification of its rules can be found in [2].

The **external, internal**, and **stochastic** choice process relations are used to describe the choice among different actions. They are respectively denoted by

$$\sum a_i \,;\, P_i \qquad \sum \tau \,;\, P_i \qquad \sum \xi_i \,;\, P_i$$

Sequence is a process relation in which two processes are consequently performed. This relation is denoted by:

$$P \,;\, Q$$

The **branch** and **switch** process relations are denoted by

No.	Meta-type	Syntax
2.1	Natural number	N
2.2	Integer	Z
2.3	Real	R
2.4	String	S
2.5	Boolean	$BL = \{\texttt{true}, \texttt{false}\}$
2.6	Byte	B
2.7	Hexadecimal	H
2.8	Pointer	$P\hat{}$
2.9	Time	$\texttt{hh} : \texttt{mm} : \texttt{ss} : \texttt{ms}$ where $\texttt{hh} \in \{0, \ldots, 23\}, \texttt{mm}, \texttt{ss} \in \{0, \ldots, 59\}, \texttt{ms} \in \{0, \ldots, 999\}$
2.10	Date	$\texttt{yy} : \texttt{MM} : \texttt{dd}$ where $\texttt{yy} \in \{0, \ldots, 99\}, \texttt{MM} \in \{1, \ldots, 12\}, \texttt{dd} \in \{1, \ldots, 31\}$
2.11	Date/Time	$\texttt{yyyy} : \texttt{MM} : \texttt{dd} : \texttt{hh} : \texttt{mm} : \texttt{ss} : \texttt{ms}$ where $\texttt{yyyy} \in \{0, \ldots, 9999\}, \texttt{MM} \in \{1, \ldots, 12\}, \texttt{dd} \in \{1, \ldots, 31\},$ $\texttt{hh} \in \{0, \ldots, 23\}, \texttt{mm}, \texttt{ss} \in \{0, \ldots, 59\}, \texttt{ms} \in \{0, \ldots, 999\}$
2.12	Run-time determinable type	RT
2.13	System architectural type	ST
2.14	Event	$@e_S$
2.15	Status	$ⓢ s_{BL}$

Table 3: Meta-types.

No.	ADT	Syntax	Designed behaviours
3.1	Stack	`Stack`:ST	Stack. (Create, Push, Pop, Clear, EmptyTest, FullTest, Release)
3.2	Record	`Record`:ST	Record. (Create, fieldUpdate, Update, FieldRetrieve, Retrieve, Release)
3.3	Array	`Array`:ST	Array. (Create, Enqueue, Serve, Clear, EmptyTest, FullTest, Release)
3.4	Queue (FIFO)	`Queue`:ST	Queue.(Create, Enqueue, Serve, Clear, EmptyTest, FullTest, Release)
3.5	Sequence	`Sequence`:ST	Sequence.(Create, Retrieve, Append, Clear, EmptyTest, FullTest, Release)
3.6	List	`List`:ST	List. (Create, FindNext, FindPrior, Findith, FindKey, Retrieve, Update, InsetAfter, InsertBefore, Delete, CurrentPos, FullTest, EmptyTest,SizeTest,Clear,Release)
3.7	Set	`Set`:ST	Set. (Create, Assign, In, Intersection, Union, Difference, Equal, Subset, Release)
3.8	File (Sequential)	`SeqFile`:ST	SeqFile. (Create, Reset, Read, Append, Clear, EndTest, Release)
3.9	File (Random)	`RandFile`:ST	RandFile. (Create, Reset, Read, Write, Clear, EndTest, Release)
3.10	Binary Tree	`BTree`:ST	BTree. (Create, TRaverse, Insert, DeleteSub, Update, Retrieve, Find, Characteristics, EmptyTest, Clear, Release)

Table 4: Abstract data types.

(CHO1) $\dfrac{}{\sum a_i;P_i \xrightarrow{a_i} P_i}$ (CHO2) $\dfrac{}{\sum \tau;P_i \xrightarrow{\tau} P_i}$ (CHO3) $\dfrac{}{\sum \xi_i;P_i \xrightarrow{\xi_i} P_i}$

(SEQ1) $\dfrac{P \xrightarrow{\alpha} P'}{P;Q \xrightarrow{\alpha} P';Q}$ (SEQ2) $\dfrac{P \xrightarrow{\surd} P'}{P;Q \xrightarrow{\tau} Q}$ (SEQ3) $\dfrac{P \xrightarrow{\xi} P'}{P;Q \xrightarrow{\xi} pP';Q}$ (SEQ4) $\dfrac{}{\text{exit} \xrightarrow{\surd} \text{stop}}$

(BRA) $\dfrac{expBL{=}\texttt{true}}{(?expBL{=}\texttt{true});P \mid (?expBL{=}\texttt{false});Q \xrightarrow{\tau} P}$ (SWI) $\dfrac{expNUM{=}i}{\mid (?expNUM{=}i);P_i \xrightarrow{\tau} P_i}$

(FOR) $\dfrac{}{\mathcal{R}_{i=1}^{n} P \xrightarrow{\tau} P;\mathcal{R}_{i=1}^{n-1} P}$ (REP) $\dfrac{}{\mathcal{R}_{\geq 1}^{expBL\neq\texttt{true}} P \xrightarrow{\tau} P;\mathcal{R}_{\geq 0}^{expBL\neq\texttt{true}} P}$

(WHI1) $\dfrac{expBL{=}\texttt{true}}{\mathcal{R}_{\geq 0}^{expBL\neq\texttt{true}} P \xrightarrow{\tau} P;\mathcal{R}_{\geq 0}^{expBL\neq\texttt{true}} P}$ (WHI2) $\dfrac{expBL{=}\texttt{false}}{\mathcal{R}_{\geq 0}^{expBL\neq\texttt{true}} P \xrightarrow{\tau} \text{exit}}$

(REC1) $\dfrac{P[X/X{:=}P] \xrightarrow{\alpha} P'}{X{:=}P \xrightarrow{\alpha} P'}$ (REC2) $\dfrac{P[X/X{:=}P] \xrightarrow{\xi} P'}{X{:=}P \xrightarrow{\xi} P'}$

(PAR1) $\dfrac{P \xrightarrow{\alpha} P'}{P \|_{tc} Q \xrightarrow{\alpha} P' \|_{tc} Q}$ (PAR2) $\dfrac{Q \xrightarrow{\alpha} Q'}{P \|_{tc} Q \xrightarrow{\alpha} P \|_{tc} Q'}$ (PAR3) $\dfrac{P \xrightarrow{a} P',\ Q \xrightarrow{b} Q',\ a*b\neq\tau}{P \|_{tc} Q \xrightarrow{a*b} P' \|_{tc} Q'}$

(PAR4) $\dfrac{P \xrightarrow{\xi} P',\ P\|_{tc}Q \not\xrightarrow{\alpha}}{P \|_{tc} Q \xrightarrow{\xi} P' \|_{tc'} \text{cond}(Q, tc'-tc)}$ (PAR5) $\dfrac{Q \xrightarrow{\xi} Q',\ P\|_{tc}Q \not\xrightarrow{\alpha}}{P \|_{tc} Q \xrightarrow{\xi} \text{cond}(P, tc'-tc) \|_{tc'} Q'}$

(CON1) $\dfrac{P \xrightarrow{\alpha} P'}{P \oint Q \xrightarrow{\alpha} P' \oint Q}$ (CON2) $\dfrac{Q \xrightarrow{\alpha} Q'}{P \oint Q \xrightarrow{\alpha} P \oint Q'}$ (CON3) $\dfrac{P \xrightarrow{a?} P',\ Q \xrightarrow{a!} Q'}{P \oint Q \xrightarrow{\tau} P' \oint Q'}$

(CON4) $\dfrac{P \xrightarrow{a!} P',\ Q \xrightarrow{a?} Q'}{P \oint Q \xrightarrow{\tau} P' \oint Q'}$ (CON5) $\dfrac{P \xrightarrow{\xi} P'}{P \oint Q \xrightarrow{\xi} P' \oint Q}$ (CON6) $\dfrac{Q \xrightarrow{\xi} Q'}{P \oint Q \xrightarrow{\xi} P \oint Q'}$

(INT1) $\dfrac{P \xrightarrow{\alpha} P'}{P \|\| Q \xrightarrow{\alpha} P' \|\| Q}$ (INT2) $\dfrac{Q \xrightarrow{\alpha} Q'}{P \|\| Q \xrightarrow{\alpha} P \|\| Q'}$ (INT3) $\dfrac{P \xrightarrow{\xi} P'}{P \|\| Q \xrightarrow{\xi} P' \|\| Q}$ (INT4) $\dfrac{Q \xrightarrow{\xi} Q'}{P \|\| Q \xrightarrow{\xi} P \|\| Q'}$

(PIPE1) $\dfrac{P \xrightarrow{\alpha} P'}{P >> Q \xrightarrow{\alpha} P';Q}$ (PIPE2) $\dfrac{P \xrightarrow{\surd} P',\ \texttt{Input}(Q){=}\texttt{Output}(P)}{P >> Q \xrightarrow{\tau} Q}$ (PIPE3) $\dfrac{P \xrightarrow{\xi} P'}{P >> Q \xrightarrow{\xi} pP' >> Q}$

(TDD) $\dfrac{t\text{system}_{\texttt{hh:mm:ss:ms}}{=}ti_{\texttt{hh:mm:ss:ms}}}{@ti_{\texttt{hh:mm:ss:ms}} \hookrightarrow P_i,\ i{\in}\{1,...,n\} \xrightarrow{\tau} P_i}$ (EDD) $\dfrac{e\text{system}{=}ei_S}{@ei_S \hookrightarrow P_i,\ i{\in}\{1,...,n\} \xrightarrow{\tau} P_i}$

(INTER1) $\dfrac{@e_S\text{captured}{=}\texttt{true}}{P \|_{tc} \bigodot(@e_S \nearrow Q \searrow \bigodot) \xrightarrow{\tau} Q;P}$ (INTER2) $\dfrac{@e_S\text{captured}{=}\texttt{false},\ P \xrightarrow{\alpha} P'}{P \|_{tc} \bigodot(@e_S \nearrow Q \searrow \bigodot) \xrightarrow{\alpha} P' \|_{tc} \bigodot(@e_S \nearrow Q \searrow \bigodot)}$

(INTER3) $\dfrac{@e_S\text{captured}{=}\texttt{false},\ P \xrightarrow{\xi} P'}{P \|_{tc} \bigodot(@e_S \nearrow Q \searrow \bigodot) \xrightarrow{\xi} P' \|_{tc} \bigodot(@e_S \nearrow Q \searrow \bigodot)}$

Table 5: Operational semantics of the process relations.

$$(?\text{expBL} = \texttt{true})\ ;\ P \mid (?\text{expBL} = \texttt{false})\ ;\ Q$$

and

$$\mid (?\text{expNUM} = i)\ ;\ P_i$$

The **FOR-DO, REPEAT**, and **WHILE-DO** process relations are denoted, respectively, by:

$$\underset{i=1}{\overset{\texttt{n}}{\mathcal{R}}}\ P \qquad \underset{\geq 1}{\overset{\texttt{expBL} \neq \texttt{true}}{\mathcal{R}}}\ P \qquad \underset{\geq 0}{\overset{\texttt{expBL} \neq \texttt{true}}{\mathcal{R}}}\ P$$

and their operational semantics are given in the rules (FOR), (REP), (WHI1) and (WHI2).

Recursion is a process relation in which the definition of a process may contain a call to itself. We will use a notation slightly different to that in RTPA

$$X := P$$

The **parallel, concurrence**, and **interleaving** process relations are denoted, respectively, by:

$$P \parallel_{tc} Q \qquad P \oint Q \qquad P \mid\mid\mid Q$$

The definition of the function $\text{cond}(P, \Delta t)$ is given in Table 6. In that table we have included, for the sake of completeness, all the cases of the function. However, the most relevant part of the definition is the one concerning choice process relations. In this case,

$$\text{cond}(P, \Delta t) = \begin{cases} \sum a_i\ ;\ \text{cond}(P_i, \Delta t) & \text{if } P = \sum a_i\ ;\ P_i \\ \sum \tau\ ;\ \text{cond}(P_i, \Delta t) & \text{if } P = \sum \tau\ ;\ P_i \\ \sum \xi_i'\ ;\ P_i & \text{if } P = \sum \xi_i\ ;\ P_i \end{cases}$$

where ξ_i' is the random variable whose probability distribution function is given as a *conditional probability*. We have that $\text{P}(\xi_i' \leq t') = \text{P}(\xi_i \leq t'' | \Delta t + t' \leq t'')$, that is, the probability of the random variable ξ_i' to finish before t' units of time have passed is equal to the probability of the original random variable ξ, to be less than or equal to t'', provided that $\Delta t + t' \geq t''$. Let us remark that the modification of the process of the parallel composition that does not perform the stochastic action is needed. This is so because, in this case, we have a multi processor single-clock system, so the time consumed by the stochastic action has to be taken into account in both sides of the parallel composition.

Pipeline is a process relation in which two processes are interconnected to each other. The second process takes the inputs from the outputs of the first one. This process relation is denoted by:

$$P >> Q$$

Time-driven dispatch is a process relation in which the i-th process is triggered by a predefined system time $ti_{\texttt{hh:mm:ss:ms}}$. It is denoted as follows:

$$@ti_{\texttt{hh:mm:ss:ms}} \hookrightarrow P_i,\ i \in \{1, \ldots, n\}$$

Event-driven dispatch is a process relation in which the i-th process is triggered by a system event $@ei_S$. It is denoted as follows:

$$@ei_S \hookrightarrow P_i,\ i \in \{1, \ldots, n\}$$

Interrupt is a process relation in which a process is temporarily held by another with higher priority. The term describing that the process P is interrupted by the process Q when the event $@e_S$ is captured at interrupt point $\odot$ will be represented by

$$P \parallel \odot(@e_S \nearrow Q \searrow \odot)$$

4. The Memorization Process

In this section we give a brief explanation on how the memorization process works. The main goal of this description is to answer the following three basic questions:

- How are memories formed? (*encoding*)
- How are memories retained? (*storage*)
- How are memories recalled? (*retrieval*)

4.1. Encoding Process

Encoding is an active process which requires selective attention to the material to be encoded. Memories may then be affected by the amount or type of attention devoted to the task of encoding the material.

There may be different levels of processing, being some of them *deeper* than others. These processes are *structural encoding* (where emphasis is placed on the physical structural characteristics of the stimulus), *phonemic encoding* (with emphasis on the sounds of the words) and *semantic encoding* (that is, emphasis on the meaning).

The main aspects of encoding fit the OAR-model in the following sense:

1. Relation: Association with other information.
2. Object: Visual imagery of the real entity or concept that can be used to add richness to the material to be remembered. Besides, it also adds more sensory modalities.
3. Attributes: To make the material personally relevant and to add more detailed information about the object.

$$\mathrm{cond}\Big(\sum a_i\,;P_i,\Delta t\Big)=\sum a_i\,;\mathrm{cond}\Big(P_i,\Delta t\Big) \qquad \mathrm{cond}\Big(\sum\ \tau\,;P_i,\Delta t\Big)=\sum\tau\,;\mathrm{cond}\Big(P_i,\Delta t\Big)$$

$$\mathrm{cond}\Big(\sum \xi_i\,;P_i,\Delta t\Big)=\sum\xi_i'\,;P_i \qquad \mathrm{cond}\Big(P\,;Q,\Delta t\Big)=\mathrm{cond}\Big(P,\Delta t\Big)\,;Q$$

$$\mathrm{cond}\Big(P>\!>Q,\Delta t\Big)=\mathrm{cond}\Big(P,\Delta t\Big)>\!>Q \qquad \mathrm{cond}\Big(\mid(?\mathrm{expNUM}=i)\,;P_i,\Delta t\Big)=\mid(?\mathrm{expNUM}=i)\,;\mathrm{cond}\Big(P_i,\Delta t\Big)$$

$$\mathrm{cond}\Big((?\mathrm{expBL}=\texttt{true})\,;P\mid(?\mathrm{expBL}=\texttt{false})\,;Q,\Delta t\Big)=(?\mathrm{expBL}=\texttt{true})\,;\mathrm{cond}\Big(P,\Delta t\Big)\mid(?\mathrm{expBL}=\texttt{false})\,;\mathrm{cond}\Big(Q,\Delta t\Big)$$

$$\mathrm{cond}\left(\mathop{\mathcal{R}}_{i=1}^{\mathtt{n}}P,\Delta t\right)=\begin{cases}\mathrm{cond}\Big(P,\Delta t\Big)\,;\mathop{\mathcal{R}}\limits_{i=1}^{\mathtt{n}-1}P & \text{if } n\geq 1\\ \mathop{\mathcal{R}}\limits_{i=1}^{\mathtt{n}}P & \text{otherwise}\end{cases}$$

$$\mathrm{cond}\left(\mathop{\mathcal{R}}_{\geq 1}^{\mathtt{expBL}\neq\mathtt{true}}P,\Delta t\right)=\mathrm{cond}\Big(P,\Delta t\Big)\,;\mathop{\mathcal{R}}_{\geq 0}^{\mathtt{expBL}\neq\mathtt{true}}P$$

$$\mathrm{cond}\left(\mathop{\mathcal{R}}_{\geq 0}^{\mathtt{expBL}\neq\mathtt{true}}P,\Delta t\right)=\begin{cases}\mathrm{cond}\Big(P,\Delta t\Big);\ \mathop{\mathcal{R}}\limits_{\geq 0}^{\mathtt{expBL}\neq\mathtt{true}}P & P \text{ if } expBL=\mathtt{true}\\ \mathop{\mathcal{R}}\limits_{\geq 0}^{\mathtt{expBL}\neq\mathtt{true}}P & \text{otherwise}\end{cases}$$

$$\mathrm{cond}\Big(X:=P,\Delta t\Big)=\mathrm{cond}\Big(P[X/X:=P],\Delta t\Big) \qquad \mathrm{cond}\Big(P\parallel_{tc}Q,\Delta t\Big)=\mathrm{cond}\Big(P,\Delta t\Big)\parallel_{tc}\mathrm{cond}\Big(Q,\Delta t\Big)$$

$$\mathrm{cond}\Big(P\oint Q,\Delta t\Big)=\mathrm{cond}\Big(P,\Delta t\Big)\oint\mathrm{cond}\Big(Q,\Delta t\Big) \qquad \mathrm{cond}\Big(P\,|||\,Q,\Delta t\Big)=\mathrm{cond}\Big(P,\Delta t\Big)\,|||\,\mathrm{cond}\Big(Q,\Delta t\Big)$$

$$\mathrm{cond}\Big(@ti_{\mathtt{hh:mm:ss:ms}}\hookrightarrow P_i,\ i\in\{1,\ldots,n\},\Delta t\Big)=@ti_{\mathtt{hh:mm:ss:ms}}\hookrightarrow P_i,\ i\in\{1,\ldots,n\}$$

$$\mathrm{cond}\Big(@ei_S\hookrightarrow P_i,\ i\in\{1,\ldots,n\},\Delta t\Big)=@ei_S\hookrightarrow\mathrm{cond}\Big(P_i,\Delta t\Big),\ i\in\{1,\ldots,n\}$$

$$\mathrm{cond}\Big(P\parallel\bigodot(@e_S\nearrow Q\searrow\bigodot),\Delta t\Big)=\mathrm{cond}\Big(P,\Delta t\Big)\parallel\bigodot(@e_S\nearrow\mathrm{cond}\Big(Q,\Delta t\Big)\searrow\bigodot)$$

Table 6: Definition of function $\mathrm{cond}(P,\Delta t)$.

4.2. Storage Process

Over the years, analogies with available technologies have been made in order to try to explain the behaviour of the memory. Nowadays, memory theories use a computer-based, or information processing, model. The most accepted model states that there are three stages of memory storage and one more for memory retrieving:

- *Sensory Store* retains the sensory image for only a small part of a second, just long enough to develop a perception. This is stored in the *Sensory Buffer Memory* (SFM). Following [12], we also consider *Action Buffer Memory* (ABM) which is used as a buffer when recovering information.
- *Short Term Memory* (STM) lasts about 20 to 30 seconds when we do not consider rehearsal of the information. On the contrary, if rehearsal is used then short term memory will last as long as the rehearsal continues. Short term memory is also limited in terms of the number of items it can hold. Its capacity is about 7 items but can be increased by *chunking*, that is, by combining similar material into units.

Let us remark that short term memory was originally perceived as a simple rehearsal buffer. However, it turns out to have a more complicated underlying process, being better modelled by using an analogy with a computer, which has the ability to store a limited amount of information in its cache RAM while performing other tasks. In other words, we can consider it as a kind of *working memory*.

- *Long Term Memory* (LTM) has been suggested to be *permanent*. However, even though no information is forgotten, we might lose the means of retrieving it.

Another interesting point regarding memory is to determine the mechanism to change the condition of a certain *memory*. In other words, how does short term memory *stuff* get into long term memory? We have to take into account the following:

- *Serial position effect*. Thus, *primacy* (i.e. first words get rehearsed more often and so that they move into long term memory) and *recency* (for instance, words at the end that are not rehearsed as often but that are still available in STM) affect long term memory.

- *Rehearsal* helps to move things into long term memory.
- According to the organizational structures of long term memory, we have also to consider:
  - Related items are usually remembered together.
  - Conceptual hierarchies are used as classification scheme to organize memories.
  - Semantic networks are less neatly organized bunches of conceptual hierarchies linked together by associations to other concepts.
  - Schemas are clusters of knowledge about an event or object abstracted from prior experience with the object. Actually, we tend to recall objects that fit our conception of the situation better than ones that do not.
  - A script is a schema which organizes our knowledge about common things or activities (if you know the script applicable to the event, you can better remember the elements of the event).

The process of storing new information in LTM is called *consolidation*.

4.3. Retrieval Process

Memory retrieval is not a random process. Once a request is generated the appropriate searching and finding processes take place. This process is triggered according to the organization structures of the LTM, while the requested information is provided via the Action Buffer Memory.

Finally, the graphic description of the memorization process is shown in Figure 1.

5. Formal description of the Memorization Process

Taking as starting point the description of the memorization process given in the previous section, we present in this section how STOPA describes in a rigorous way this cognitive process of the brain.

// PN**N** =1
MemorizationProcess (I:: ThePerceptionS;
O:: OAR (ThePerception**S**)**ST**, Lmemorization**N**)
{// PNN = 2
o**S** := ThePerception**S**
→ (Scope**S** := Objects**S** ↪
Search (I:: oS; Scope**S**; O::$\{o_1, o_2, \ldots, o_n\}$) // PN**N** =3
|| Scope**S** := Attributes**S** ↪

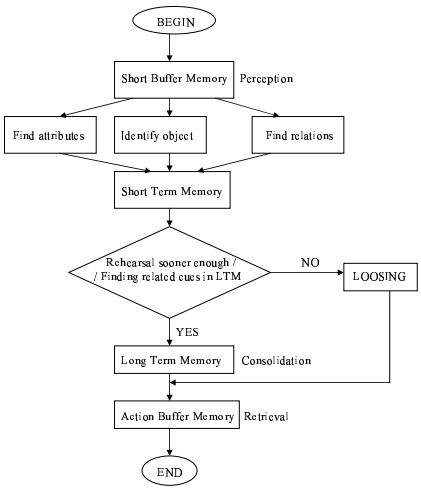

Figure 1: The model of the memorization process

Search (I:: oS; Scope**S**; A::$\{a_1, a_2, \ldots, a_m\}$) // PN**N** =4
|| Scope**S** := Relations**S** ↪
Search (I:: oS; Scope**S**; R::$\{r_1, r_2, \ldots, r_t\}$) // PN**N** =5
)
→ EncodingSTM (I:: OAR(o**S**);
O:: OAR(o**S**)) { } // PN**N** = 6
→ (? (time in REHEARSAL)=`true` // PN**N** = 7
→ EncodingLTM (I:: OAR(o**S**);
O:: OAR(o**S**)) { } // PN**N** = 8
→ PL1**S**
| ? (time in REHEARSAL)=`false` // PN**N** =10
→ LOOSING (I:: OAR(o**S**); O::)
→ PL1**S**
)
→ PL1**S** ↪ DecodingABM (I:: OAR(o**S**);
O:: TheInformation(**ST**)) { } // PN**N** =9
}// PNN =11

6. Conclusions

In this paper we have used the algebraic formalism STOPA, a process algebra with very general stochastic time. This feature allows us to consider time which is not given by a fix amount but it depends on general probability distribution function. The good characteristics of this language has been pointed out by the formal representation of the cognitive process of memorization, here provided.

We contemplate several lines for future work:

- Developing this formalism by means of the definition of a semantics over it, which should allow us to compare the behaviour of, *a priori*, different STOPAprocesses.
- Using some other graphical formalism like Petri Nets or Automaton for this task of representing cognitive processes.
- Studying in a detailed way some other cognitive processes and systems, for better testing the capabilities of STOPA.

References

[1] D. Cazorla, F. Cuartero, V. Valero, F. L. Pelayo, and J. Pardo. Algebraic theory of probabilistic and nondeterministic processes. *Journal of Logic and Algebraic Programming*, 55(1–2):57–103, 2003.

[2] N. López, M. Núñez, and F. L. Pelayo. Stopa: A stochastic process algebra for the formal representation of cognitive systems. In *3nd IEEE Int. Conf. on Cognitive Informatics*, pages 1–1. IEEE Computer Society Press, 2004.

[3] N. López, M. Núñez, and F. Rubio. An integrated framework for the analysis of asynchronous communicating stochastic processes, 2004. To appear in *Formal Aspects of Computing*.

[4] M. Núñez. Algebraic theory of probabilistic processes. *Journal of Logic and Algebraic Programming*, 56(1–2):117–177, 2003.

[5] M. Núñez, D. de Frutos, and L. Llana. Acceptance trees for probabilistic processes. In *CONCUR'95, LNCS 962*, pages 249–263. Springer, 1995.

[6] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[7] R. Solso, editor. *Mind and brain science in the 21st century*. MIT Press, 1999.

[8] L. Squire, B. Knowlton, and G. Musen. The structure and organization of memory. *Annual Review of Psychology*, 44:453–459, 1993.

[9] Y. Wang. On cognitive informatics. In *1st IEEE Int. Conf. on Cognitive Informatics*, pages 34–42. IEEE Computer Society Press, 2002.

[10] Y. Wang. The Real Time Process Algebra (RTPA). *Annals of Software Engineering*, 14:235–274, 2002.

[11] Y. Wang. Using process algebra to describe human and software behaviors. *Brain and Mind*, 4:199–213, 2003.

[12] Y. Wang and Y. Wang. Cognitive models of the brain. In *1st IEEE Int. Conf. on Cognitive Informatics*, pages 259–269. IEEE Computer Society Press, 2002.

[13] S. Zhongzhi. *Neural Computing*. Electronic Industry Publishers, 1994.

[14] S. Zhongzhi and Y. Zhihua. *Cognitive Science and Computer*. Popularization Publishers, 1999.