Specification and Performance of the MPEG-2 Video Encoder by Using the Stochastic Process Algebra: ROSA*

Fernando L. Pelayo, Fernando Cuartero, Valentín Valero, Diego Cazorla and Teresa Olivares †

Abstract

In this paper we present the specification of the MPEG-2 algorithm for video encoding, by using the Stochastic Process Algebra **ROSA** (**R**easoning **O**n **S**tochastic **A**lgebras). This process algebra is a very general framework for describing and analysing concurrent systems. Thus, we also study the temporal behaviour of the algorithm, by using a performance evaluation algorithm, which is also presented in this paper.

1 Introduction

In the eighties, fundamentals in process algebras were firmly established, but the proliferation of distributed systems, real-time systems and fault-tolerant systems has generated a great interest in the definition of timed and probabilistic extensions of these models.

A first approach to incorporate probabilities in process algebra can be found in [2]. A pioneer work on a probabilistic extension of CCS is PCCS [7], where the non-deterministic choice is replaced by a probabilistic choice operator.

Later, in [8] is defined *Timed Probabilistic Calculus of Communicating Systems* (TPCCS), which extends CCS with both probabilities and time.

There are also some extensions of CSP [4, 5] but in general, the underlying idea in all these probabilistic models is to replace a non-deterministic behaviour by a probabilistic one, in which every possible behaviour of the system is quantified by means of a probability. But obviously, this requires a quantitative knowledge of the possible behaviours of the modelled system. Unfortunately, it is not always possible to have such a knowledge, although it can be sometimes estimated [12].

Thus, part of our recent work has been to define a model that combines both kind of choices, non-deterministic and probabilistic ones. PNAL is an algebraic language which extends Hennessy's EPL by means of a probabilistic choice, so, in [3] this language is presented and three semantics for it are given. The language we use, **ROSA**, has some functional similarities with PNAL.

But, it is also desirable to take into account performance aspects in the design of concurrent systems. It is true that often efficiency considerations are forgotten until the system works properly, i.e., once it has been fully tested.

*This work has been supported by the Ministry of Education of Spain under the CICYT project "Performance Evaluation of Distributed Systems",TIC2000-0701-C02-02.

†Escuela Politécnica Superior de Albacete. University of Castilla-La Mancha. Campus Universitario s/n. 02071, Albacete, Spain. Email: {fpelayo, fernando, valentin, dcazorla, teresa}@info-ab.uclm.es

However, it becomes obvious that there are some systems (real-time systems) where we need to know a priori if the system will fulfil with its restrictions. Then, a model for describing concurrent systems should also include some capabilities to determine efficiency considerations.

Some stochastic process algebras are presented in [10, 6, 9, 1] considering these capabilities. We can find in [1] the language EMPA *Extended Markovian Process Algebra*, which has two kinds of actions like our model, and the *race policy* is applied to decide the actions to be executed. However, a syntactic restriction must be introduced to define the behaviours of synchronizations, because one of the actions involved in a synchronization must be always a passive one, which makes it hard to be able to model a synchronization between two actions which have both a certain temporal cost. Even more, a problem which could appear in this language is that higher priority actions always prevent the execution of lower level ones. For instance, if we consider a server which offers two services a and b, having a a greater priority than b, we find that a client that only wants to execute a b could not be served, because the EMPA process $(< a, \infty_{l,w} > . \underline{0} + < b, \infty_{l',w'} > . \underline{0}) \;||_{\{a\}} < b, * > . \underline{0}$, taking $l > l'$, cannot execute any actions at all.

ROSA[14], follows some ideas presented in [13], so, it does not impose any syntactical restrictions on the components of a parallel operator; thus, the parameter λ associated to any action does not limit its capabilities for a synchronization.

The paper is structured as follows: in Section 2 the syntax of **ROSA** is defined, the operational semantics for this language is described in Section 3, and in Section 4 the MPEG-2 Video Encoder Algorithm is described. The performance evaluation algorithm is shown in Section 5 and Section 6 presents as case study the analysis of MPEG-2. Finally, some conclusions are presented in Section 7.

2 Syntax of the Language

Let $\Delta = \{\tau, a, b, c, ..\}$ be a set of action types, where τ stands for an internal action type, and the remaining ones are all visible.
We will denote by letters p, q, r probabilities and by greek letters time parameters for actions.

Terms of **ROSA** are defined by:

$$P ::= \mathbf{0} \mid X \mid a.P \mid \langle a, \lambda\rangle.P \mid P \oplus P \mid P + P \mid P {\oplus_r} P \mid P||_A P \mid recX : P \mid P\backslash A$$

where $r \in (0, 1)$, $\lambda \in \mathbb{R}^+ - \{0\}$, $A \subseteq (\Delta - \{\tau\})$, $a \in \Delta$ and P is a process of **ROSA**.

The standard operators are interpreted as usual in classical process algebras, i.e., $a.P$ stands for the prefix operator, $\oplus$ and $+$ are respectively the internal and the external choices, $||_A$ is the parallel operator (synchronizing on actions with types in A), $recX : P$ is the recursion operator, and $P\backslash A$ is the hiding operator (actions with types in A are hidden).

The informal interpretation of the new operators follows:

- $\langle a, \lambda\rangle.P$, which stands for a timed prefix operator, where the action labelled with a has a duration following a Negative Exponential random distribution with parameter λ.
  Let us observe that now $a.P$ is a particular case of this operator, when we take $\lambda = \infty$, which means that $\mathbf{E}[Exp[\infty]] = 0$, i.e. this action has an average duration 0, which means that it is immediate.

(S1a) $$\frac{}{a.P \xrightarrow{1,a,\infty} P}$$

(S1b) $$\frac{}{\langle a,\lambda\rangle.P \xrightarrow{1,a,\lambda} P}$$

(S2a) $$\frac{}{P \oplus_p Q \xrightarrow{p,\tau,\infty} P}$$

(S2b) $$\frac{}{P \oplus_p Q \xrightarrow{1-p,\tau,\infty} Q}$$

(S3a) $$\frac{}{P \oplus Q \xrightarrow{\tau,\infty} P}$$

(S3b) $$\frac{}{P \oplus Q \xrightarrow{\tau,\infty} Q}$$

(S3c) $$\frac{P \xrightarrow{\tau,\infty} P'}{P + Q \xrightarrow{\tau,\infty} P' + Q}$$

(S3d) $$\frac{Q \xrightarrow{\tau,\infty} Q'}{P + Q \xrightarrow{\tau,\infty} P + Q'}$$

(S3e) $$\frac{P \xrightarrow{\tau,\infty} P'}{P||_A Q \xrightarrow{\tau,\infty} P'||_A Q}$$

(S3f) $$\frac{Q \xrightarrow{\tau,\infty} Q'}{P||_A Q \xrightarrow{\tau,\infty} P||_A Q'}$$

(S4a) $$\frac{P \xrightarrow{q,a,\lambda} P' \wedge a \notin \mathbf{Type}[\mathbf{D}[Q]]}{P + Q \xrightarrow{q,a,\lambda} P'}$$

(S4b) $$\frac{Q \xrightarrow{r,a,\lambda} Q' \wedge a \notin \mathbf{Type}[\mathbf{D}[P]]}{P + Q \xrightarrow{r,a,\lambda} Q'}$$

(S4c) $$\frac{P \xrightarrow{q,a,\infty} P' \wedge Q \xrightarrow{r,a,\infty} Q'}{P + Q \xrightarrow{\tau,\infty} P \xrightarrow{q,a,\infty} P'}$$

(S4d) $$\frac{P \xrightarrow{q,a,\infty} P' \wedge Q \xrightarrow{r,a,\infty} Q'}{P + Q \xrightarrow{\tau,\infty} Q \xrightarrow{r,a,\infty} Q'}$$

(S4e) $$\frac{P \xrightarrow{q,a,\lambda_1} P' \wedge Q \xrightarrow{r,a,\lambda_2} Q' \wedge \lambda_2 \neq \infty}{P + Q \xrightarrow{\tau,\infty} P \xrightarrow{q\cdot r,a,\lambda_1+\lambda_2} P'}$$

(S4f) $$\frac{P \xrightarrow{q,a,\lambda_1} P' \wedge Q \xrightarrow{r,a,\lambda_2} Q' \wedge \lambda_1 \neq \infty}{P + Q \xrightarrow{\tau,\infty} Q \xrightarrow{q\cdot r,a,\lambda_1+\lambda_2} Q'}$$

(S4g) $$\frac{P \xrightarrow{q,a,\lambda_1} P' \wedge Q \xrightarrow{r,a,\lambda_2} Q' \wedge \exists \lambda_i \neq \infty}{P + Q \xrightarrow{q\cdot(1-r),a,\lambda_1} P'}$$

(S4h) $$\frac{P \xrightarrow{q,a,\lambda_1} P' \wedge Q \xrightarrow{r,a,\lambda_2} Q' \wedge \exists \lambda_i \neq \infty}{P + Q \xrightarrow{(1-q)\cdot r,a,\lambda_2} Q'}$$

(S5a) $$\frac{P \xrightarrow{q,a,\lambda} P' \wedge a \notin \mathbf{Type}[\mathbf{D}[Q]] \cup A}{P||_A Q \xrightarrow{q,a,\lambda} P'||_A Q}$$

(S5b) $$\frac{Q \xrightarrow{r,a,\lambda} Q' \wedge a \notin \mathbf{Type}[\mathbf{D}[P]] \cup A}{P||_A Q \xrightarrow{r,a,\lambda} P||_A Q'}$$

(S5c) $$\frac{P \xrightarrow{q,a,\infty} P' \wedge Q \xrightarrow{r,a,\infty} Q' \wedge a \notin A}{P||_A Q \xrightarrow{\tau,\infty} P||_A Q \xrightarrow{q,a,\infty} P'||_A Q}$$

(S5d) $$\frac{P \xrightarrow{q,a,\infty} P' \wedge Q \xrightarrow{r,a,\infty} Q' \wedge a \notin A}{P||_A Q \xrightarrow{\tau,\infty} P||_A Q \xrightarrow{r,a,\infty} P||_A Q'}$$

(S5e) $$\frac{P \xrightarrow{q,a,\lambda_1} P' \wedge Q \xrightarrow{r,a,\lambda_2} Q' \wedge a \notin A \wedge \lambda_2 \neq \infty}{P||_A Q \xrightarrow{\tau,\infty} P||_A Q \xrightarrow{q\cdot r,a,\lambda_1+\lambda_2} P'||_A Q}$$

(S5f) $$\frac{P \xrightarrow{q,a,\lambda_1} P' \wedge Q \xrightarrow{r,a,\lambda_2} Q' \wedge a \notin A \wedge \lambda_1 \neq \infty}{P||_A Q \xrightarrow{\tau,\infty} P||_A Q \xrightarrow{q\cdot r,a,\lambda_1+\lambda_2} P||_A Q'}$$

(S5g) $$\frac{P \xrightarrow{q,a,\lambda_1} P' \wedge Q \xrightarrow{r,a,\lambda_2} Q' \wedge a \notin A \wedge \exists \lambda_i \neq \infty}{P||_A Q \xrightarrow{q\cdot(1-r),a,\lambda_1} P'||_A Q}$$

(S5h) $$\frac{P \xrightarrow{q,a,\lambda_1} P' \wedge Q \xrightarrow{r,a,\lambda_2} Q' \wedge a \notin A \wedge \exists \lambda_i \neq \infty}{P||_A Q \xrightarrow{(1-q)\cdot r,a,\lambda_2} P||_A Q'}$$

(S5i) $$\frac{P \xrightarrow{q,a,\lambda_1} P' \wedge Q \xrightarrow{r,a,\lambda_2} Q' \wedge a \in A}{P||_A Q \xrightarrow{q\cdot r,a,\mathbf{min}[\{\lambda_1,\lambda_2\}]} P'||_A Q'}$$

(S6a) $$\frac{P[recX : P/X] \xrightarrow{q,a,\lambda} P'}{recX : P \xrightarrow{q,a,\lambda} P'}$$

(S7a) $$\frac{P \xrightarrow{q,a,\lambda} P' \wedge a \notin A}{P\backslash A \xrightarrow{q,a,\lambda} P'\backslash A}$$

(S7b) $$\frac{P \xrightarrow{q,a,\lambda} P' \wedge a \in A}{P\backslash A \xrightarrow{q,\tau,\lambda} P'\backslash A}$$

Table 1: Operational Semantics for **ROSA**

- $P \oplus_r Q$ denotes the probabilistic choice with the classical generative meaning: with probability r the process behaves like P, and with probability $1 - r$ it behaves like Q (the external environment has no influence in this choice).

3 Operational Semantics

The operational semantics is defined in the Plotkin and Milner´s style (see Table 1), by using a labelled transition system with two kinds of transitions:

- Probabilistic transitions:
$$P \xrightarrow{r,a,\lambda} Q$$
where r is a probability, a is an action type and λ is the parameter of an Exponential distribution modelling the action duration. With this rule we represent that the process P can evolve with probability r, by executing the action labelled by a, with a duration described by an Exponential distribution with parameter λ.

- Non-deterministic transitions:
$$P \xrightarrow{\tau,\infty} Q$$
which represent the internal decisions that the system makes for resolving the non-deterministic choices. These evolutions do not take any time at all.

Rules S1, S2, S3, S6 and S7 are quite obvious, just note that rules S2a and S2b establish the precedence of the probabilistic choice over the internal one, and rules S3c to S3f impose the precedence of the internal choice over the external choice and the parallel operators.

Rules S4 capture the semantics of the external choice; concretely S4a and S4b represent the evolution by executing an action that only is at the disposal of one component. When an action can be executed by both arguments, rules S4c to S4h establish the evolution of $P + Q$, by distinguishing all the possible cases according to the duration of the involved actions. If the action is immediate for both components (it has zero duration), then the choice is resolved in a non-deterministic way, which is captured by rules S4c and S4d. These non-deterministic evolutions can appear when solving internal choices, as usual, but, when they capture the non-determinism associated with the competition between two arguments for executing an action, they will appear as the first evolution of some "special" rules (S4c - S4f and S5c - S5f) followed by the corresponding execution of each action. Whereas if the action is not immediate for at least one of the components, we have that the evolution will be made either in a probabilistic way, or, firstly in a non-deterministic way and in a probabilistic way later, according to the following cases:

- If both components cooperate for executing the action, this is, both arguments try to execute the action, then the quickest one will execute it (rules S4e and S4f), so that a non-deterministic evolution takes place and afterwards a probabilistic one, which takes as new parameter the sum of the parameters of that action for each component, which means that the action will have a shorter duration, and the probability for this evolution is the product of the probabilities for each component. The fact of accelerating the execution of the actions when cooperating, and the urgent character of immediate ones is called *race policy*.

- Otherwise, with rules S4g and S4h we capture the evolution when only one of the components tries to execute the action. In this case we preserve the duration of the action corresponding to the component that executes it, but the probability for this case is the product of q and $1 - r$, where r is the probability to execute the action by the component which does not cooperate in its execution.

Rules S5 describe the behaviour for the parallel composition operator, we can see that they are very similar to those described for the external choice operation. We just need to mention rule S5i, which captures the evolution of $P||_A Q$ by executing an action belonging to the synchronization set A. In this case we see that the new duration is obtained from an Exponential distribution with parameter $\mathbf{min}[\{\lambda_1, \lambda_2\}]$, which means that the slowest one imposes its duration.

We have used two auxiliary functions in the operational semantics:

- **D** is a function defined on the set of **ROSA** processes which associates to every process its multiset of available actions.
- **Type** is a function defined on the set of multisets of actions which associates to every multiset, the set consisting of their action types.

4 The MPEG-2 Digital Video Coding Standard

The ISO/IEC 13818–2 standard [11], commonly known as MPEG–2, is a standard intended for a wide range of applications, including Video–on–Demand (VoD), High Definition TV (HDTV) and video communications using broadband networks.

The MPEG standards were designed with two requirements in mind, namely, need for high compression, and need for random access capability.

The MPEG digital video coding techniques are statistical in nature. Video sequences usually contain statistical redundancies in both temporal and spatial directions. The basic statistical property upon which MPEG compression techniques rely is inter–pixel region correlation. The contents of a particular pixel region can be predicted from nearby pixel regions within the same frame (intra–frame coding) or from pixel regions of a nearby frame (inter–frame coding).

Perhaps the ideal method for reducing temporal redundancy is one that tracks every pixel from frame to frame. However, this extensive search is computationally expensive. Under the MPEG standards, this search is performed by tracking the information within 16×16 pixels regions, called macroblocks. Given two contiguous frames, $frame(t)$ and $frame(t-1)$, for each macroblock in $frame(t)$, the coder determines the best matching macroblock in $frame(t-1)$ and calculates the translation of the picture macroblock between frames, obtaining the motion vector. Using the corresponding macroblock from $frame(t-1)$, the temporal redundancy reduction processor generates a representation for $frame(t)$ that contains only the motion vector and the prediction error (changes between the two frames). This technique is called *motion compensated prediction*.

In order to reduce spatial redundancy a DCT (Discrete Cosine Transform) a coding method is used; Low DCT coefficients relate to low spatial frequencies within image blocks and high DCT coefficients to higher frequencies. Since the human viewer is more sensitive to reconstruction error related to low spatial frequencies than to high frequencies, a frequency adaptive weighting (quantization) of the coefficients according to the human visual perception is often employed to improve the visual quality of the decoded images.

The combination of the two techniques described above (temporal motion compensated prediction and transform domain coding) are the key elements of the MPEG coding.

In order to achieve the requirement of random access and high compression, the MPEG–2 standard specifies three types of compressed video frames: I pictures, P pictures and B pictures. I pictures (intracoded pictures) are coded with no reference to other frames, exploiting only spatial correlation in a frame. P pictures (predictive coded pictures) are coded by using motion compensated prediction of a previous I or P picture. Finally, B pictures (bidirectionally–predictive coded pictures) are obtained by motion compensation by using past and future reference frames (I or P pictures).

A group of consecutive I, P and B pictures form a structure called Group of Pictures (GoP). A video sequence may be seen, then, as a sequence of GoPs.

The block diagram of the MPEG encoder is depicted in Figure 1. In order to understand how the MPEG–2 encoder works, we will consider a typical GoP consisting on the frames IBBP. Despite B pictures appear before P pictures, the coding order is IPBB because B pictures require both past and future frames as references.

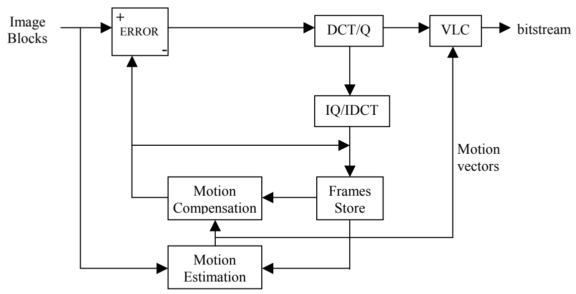

Figure 1: Block diagram of the MPEG–2 encoder

The first frame in a GoP (I picture) is encoded in intra mode without reference to any past or future frames. At the encoder the DCT is applied to each macroblock and then is uniformly quantized (Q). After quantization, it is coded using a variable length code (VLC) and send to the output buffer. At the same time the reconstruction (IQ) of all non–zero DCT coefficients belonging to one macroblock and subsequent inverse DCT (IDCT) give us a compresses I picture which is stored temporarily in the Frames Store (FS).

If the input is coded either as P picture or B picture, then the encoder does not code the picture macroblocks directly. Instead, it codes the prediction errors and the motion vectors. With P pictures, for each macroblock in the current picture, the motion estimation gives us the coordinates of the macroblock in the I picture that best matches its characteristics and thus, the motion vector may be calculated. The motion compensated prediction error is obtained by subtracting each pixel in a macroblock with its motion shifted counterpart in the previous frame. The prediction error and the motion vectors are coded (VLC) and send to the output buffer. As in the previuos case, a compressed P picture is stored in the Frames Store.

With B pictures, the motion estimation process is performed two times: for a past picture (I picture in this case), and for a future picture (P picture). Prediction error and both motion vectors for each macroblock are coded (VLC) and send to the output buffer. It is not necessary to store in FS the compressed B picture.

5 Performance Evaluation in ROSA

Starting from the operational semantics, we can construct for each process a *transition graph*, which can be used for performance evaluation studies, according to the following steps:

1. To begin with, non-deterministic transitions are eliminated from this *transition graph*, and we split the graph into several new graphs, according to Fig.2. Thus, we obtain a set of *deterministic transition graphs* which have only transitions labelled with three parameters.

2. Now, from each one of these *deterministic transition graphs*, we relabel every transition: $\xrightarrow{q,a,\lambda}$ with $\xrightarrow{q,1/\lambda}$, thus obtaining the so called *deterministic temporal graph*, which only has information about both the probability and the average time associated to every transition (Fig.2).

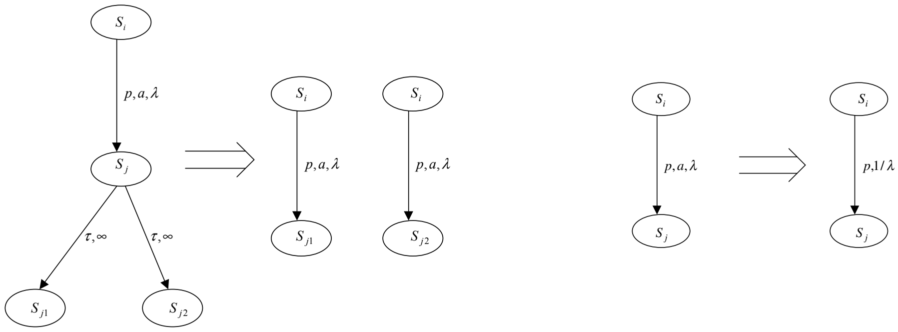

Figure 2: Evaluation Algorithm - Phases 1 and 2

3. Next, for each graph, we eliminate those transitions having zero temporal cost by joining the nodes connected by that transition in a single one, labelled either by S_i or by S_j as Fig.3 shows. The probability of the outgoing edges are weighted by the probability of the eliminated one.

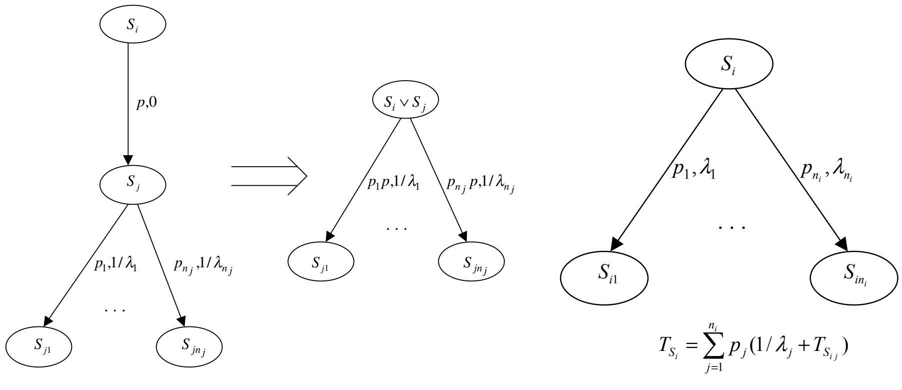

Figure 3: Evaluation Algorithm - Phases 3 and 4

4. Once we have carried out all these steps, we get a set of *deterministic reduced temporal graphs*. For each one we can compute the time required to reach a concrete state S_F from the initial one S_0 (let us observe that each *deterministic reduced temporal graph* only has one initial state, which will be drawn by a double oval, and one final state, represented by a double rectangle).

   Then, for each *deterministic reduced temporal graph* we denote by T_{S_i} the expected time to evolve from the state S_i to the final state (S_F).

   We have $(T_{S_i} = \sum_{j=1}^{n_i} p_j(1/\lambda_j + T_{S_{ij}}))$, being n_i the number of outgoing edges from the state S_i, and the label of the edge connecting the state S_i with S_{ij} is $(p_j, 1/\lambda_j)$ as we can see in Fig.3. As a consequence, the average time to reach the final state from the starting one will be T_{S_0}, which could be computed by unfolding this recurrent equation.

At this point, we have calculated as many average times as different non-deterministic ways exist (according to the obtained *deterministic reduced temporal graphs*) to evolve from the initial state to the final one. Note that this number is exponential on the number of non-deterministic choices.

The highest and the lowest average time thus obtained provide us with time information concerning the reachability of the state S_F from the state S_0.

6 Specification and Analysis of the Sequential MPEG-2 Encoding Algorithm

We study the encoding process of a GoP formed by four consecutive images (frames) which are considered as I, B, B, and P images respectively, altough the order for the encoding will be IPBB because of the particularities described in section 4.

I frame:
$I \equiv \langle DCT\_Q_I, \alpha_1\rangle.(\langle VLC_I, \alpha_2\rangle.out \,||_A\, \langle IQ\_IDCT_I, \alpha_3\rangle.\langle FS_I, \alpha_4\rangle.$
$.EST_P.COMP_P.\Theta)$
P frame:
$P \equiv \langle EST_P, \gamma_1\rangle.\langle COMP_P, \gamma_2\rangle.\langle ERROR_P, \gamma_3\rangle.\langle DCT\_Q, \gamma_4\rangle.(\langle VLC_P, \gamma_5\rangle.$
$.out \,||_A\, \langle IQ\_IDCT, \gamma_6\rangle.\langle FS_P, \gamma_7\rangle.\Theta)$
B_i frames:
$B_i \equiv \langle EST_{B_i}, \beta_1\rangle.\langle COMP_{B_i}, \beta_2\rangle.\langle ERROR_{B_i}, \beta_3\rangle.\langle DCT\_Q, \beta_4\rangle.\langle VLC_{B_i}, \beta_5\rangle.out$

$\Theta \equiv (EST_{B_1}.COMP_{B_1} \,||_A\, EST_{B_2}.COMP_{B_2})$

MPEG-2 Encoding Algorithm:

$$MPEG\text{-}2 \equiv I \,||_A\, P \,||_A\, B_1 \,||_A\, B_2$$
$$A = \{EST_P, EST_{B_1}, EST_{B_2}, COMP_P, COMP_{B_1}, COMP_{B_2}\}$$

Notice that in this algorithm there is not probabilistic behaviour, so that we omit this information for the whole this study.

Next figure shows the *deterministic transition graph* corresponding to the MPEG-2 video encoding algorithm, where the initial, the final and those states in which a frame has been encoded are marked with square boxes:

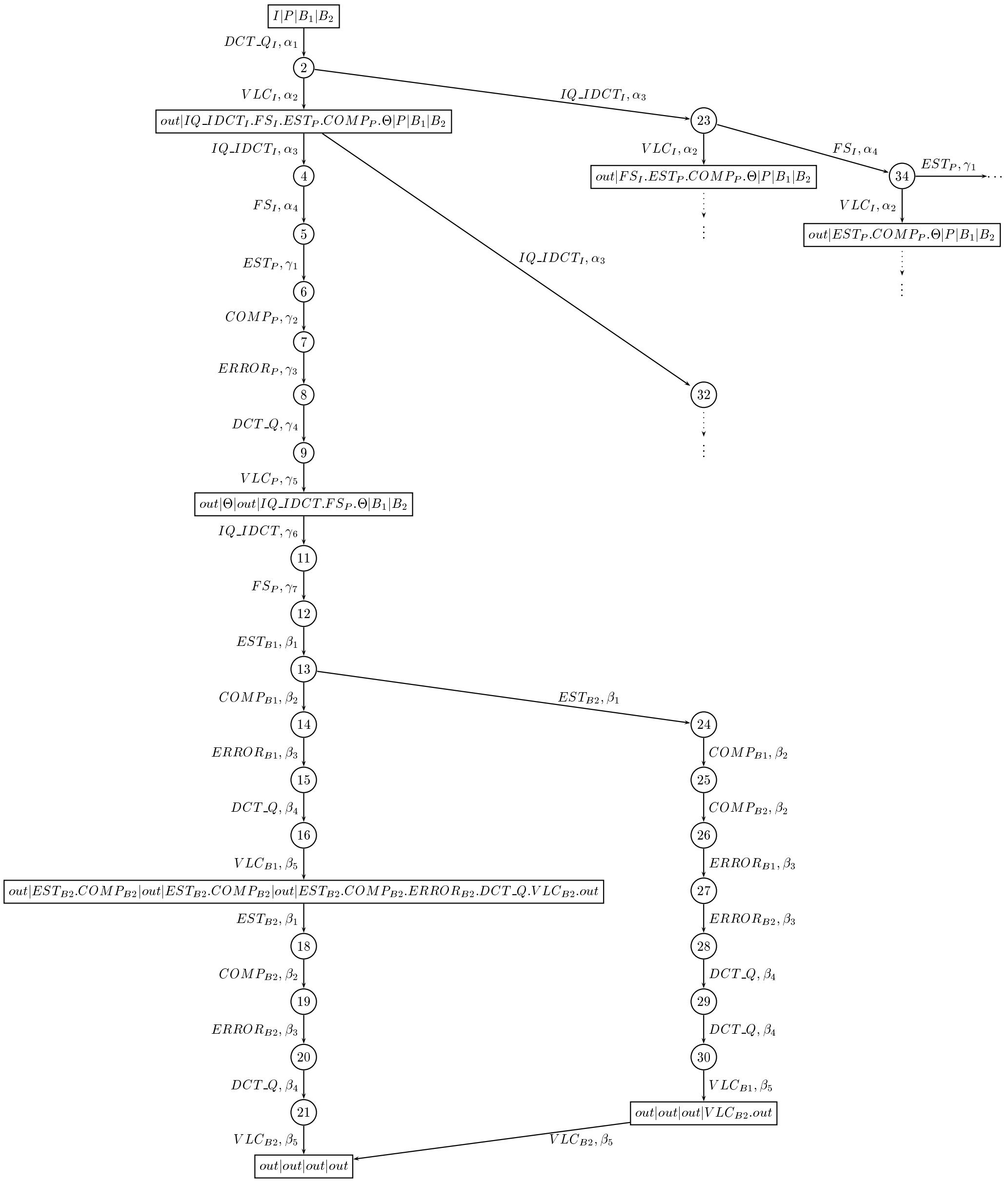

Figure 4: *Deterministic Transition Graph of MPEG-2*

We can observe in Figure 4 that from state 13 there are two possible evolutions which correspond with the traces of both the quickest and the slowest ways for encoding the B_1-image.

Figure 5 shows these two possible evolutions in two separated graphs.

We can compute the average time to complete the codification of the GoP in both cases, obtaining the same value, T:

$$\mathrm{T} = \frac{1}{\alpha_1} + \frac{1}{\alpha_2} + \frac{1}{\alpha_3} + \frac{1}{\gamma_1} + \frac{1}{\gamma_2} + \frac{1}{\gamma_4} + \frac{1}{\gamma_5} + \frac{1}{\gamma_6} + \frac{2}{\beta_1} + \frac{2}{\beta_2} + \frac{2}{\beta_4} + \frac{2}{\beta_5}$$

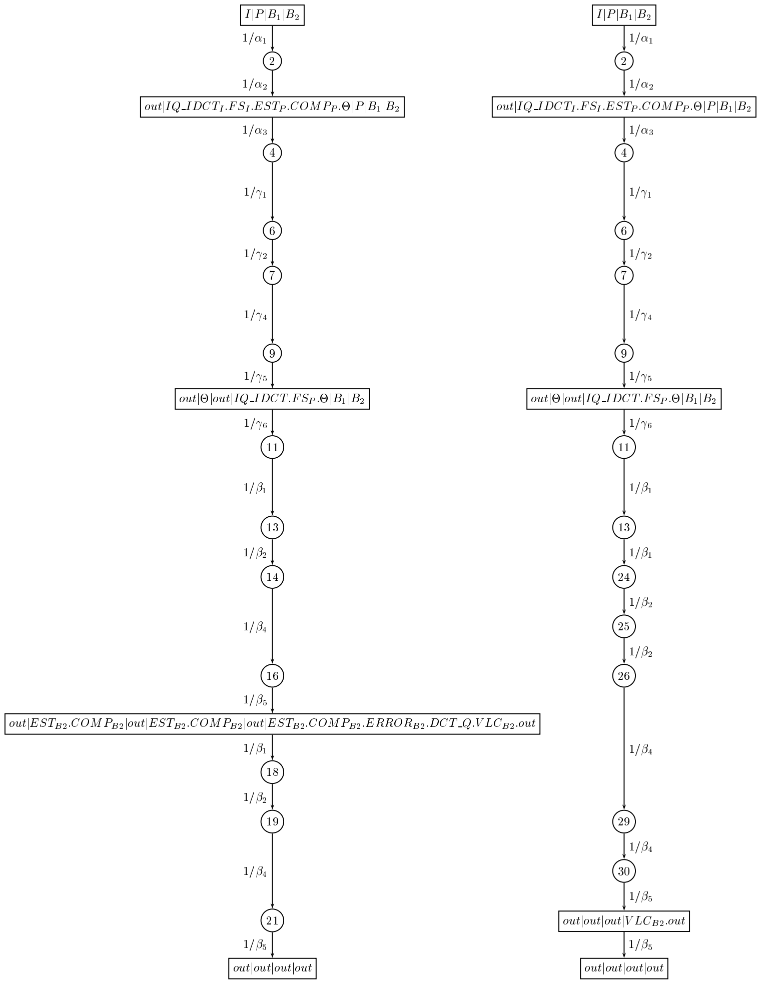

Figure 5: *The Quickest and the Slowest encoding traces*

Also if we compute the average times for encoding the I-image and the P-image, the values that we compute for both graphs are the same:

$$\mathrm{T_I} = \frac{1}{\alpha_1} + \frac{1}{\alpha_2}$$

$$\mathrm{T_{IP}} = \frac{1}{\alpha_1} + \frac{1}{\alpha_2} + \frac{1}{\alpha_3} + \frac{1}{\gamma_1} + \frac{1}{\gamma_2} + \frac{1}{\gamma_4} + \frac{1}{\gamma_5}$$

However, when we compute the average time for encoding the B_1-image we can obtain some different results, then we have selected those corresponding with the quickest and the slowest average encoding times:

$$\mathrm{T_{IPB_1Q}} = \frac{1}{\alpha_1} + \frac{1}{\alpha_2} + \frac{1}{\alpha_3} + \frac{1}{\gamma_1} + \frac{1}{\gamma_2} + \frac{1}{\gamma_4} + \frac{1}{\gamma_5} + \frac{1}{\gamma_6} + \frac{1}{\beta_1} + \frac{1}{\beta_2} + \frac{1}{\beta_4} + \frac{1}{\beta_5}$$

$$\mathrm{T_{IPB_1S}} = \frac{1}{\alpha_1} + \frac{1}{\alpha_2} + \frac{1}{\alpha_3} + \frac{1}{\gamma_1} + \frac{1}{\gamma_2} + \frac{1}{\gamma_4} + \frac{1}{\gamma_5} + \frac{1}{\gamma_6} + \frac{2}{\beta_1} + \frac{2}{\beta_2} + \frac{2}{\beta_4} + \frac{1}{\beta_5}$$

6.1 A Real Codification Experiment

We have encoded a sequence in order to compare with real measurements the theoretical results so obtained, specially those results concerning the time to encode the first B-frame.

The experiment was repeated three times. The results being reported are therefore the average of three trials. During the trials, not other operations were taken place in our experimental setup. We have realized the experiments with a video sequence representative of different levels of motion, the "Composed" sequence (format PAL CCIR601, 720x576 pixels). We have used a completely software-based MPEG-2 video encoder derived from that developed in Berkeley which is freely available in the MPEG Home Page (http://drogo.cslet.stet.it/mpeg).

We have marked some "points" in the source code which correspond with the beginning and the end of the elementary actions described in the specification, in order to obtain some time measurements for estimating the temporal parameters of the Negative Exponentials which model the durations associated with those elementary actions of the specification.

The platform used for the experiments was a Pentium II - 350MHz processor with 64MB RAM.

The estimated rates are:

$\alpha_1 = 1.681$	$\alpha_2 = 3.089$	$\alpha_3 = 1.486$	$\gamma_1 = 0.847$
$\gamma_2 = 31.658$	$\gamma_4 = 1.920$	$\gamma_5 = 2.600$	$\gamma_6 = 1.942$
$\beta_1 = 0.415$	$\beta_2 = 20.626$	$\beta_4 = 1.920$	$\beta_5 = 2.613$

The remainder parameters are ∞, i.e. they correspond with immediate actions.

Once we have estimated the time parameters we can compare the theoretical results obtained with our language **ROSA**, and some measures obtained in real codifications with the described scenario.

Average time to encode the I-image, by **ROSA**: 0.918613s

Real measures: 0.909285s, 0.953689s, 0.893104s

Average time to encode the P-image, by **ROSA**: 3.70923s

Real measures: 3.700998s, 3.738053s, 3.688503s

Average time to encode the GoP, by **ROSA**: 10.9475s

Real measures: 11.549177s, 10.793488s, 11.067721s

ROSA allows to compute the bounds for the average time to encode the B_1-image: 7.58582s - 10.5648s

We can conclude that by using two processors the necessary time for encoding a GoP like the considered, can be reduced about a 25 per cent, and in bigger GoPs the reduction factor increases because of the possibility of real parallel codification of B-images which means that both B-images can be coded at once.

7 Conclusions and Future Work

It has been presented an algebraic language, **ROSA**, for the modelling and the analysis of concurrent systems with probabilities, non-determinism and time.

Then, **ROSA** has been used in order to study some performance features of the MPEG-2 video encoding algorithm as bounds for the time necessary to have codified each type of image.

Our current work is mainly focused on analyzing another version of this algorithm which contemplates the possibility of system failure as well as on analyzing and comparing two different parallel versions of the MPEG-2 encoding algorithm: Spatial Parallelism vs Temporal Parallelism.

References

[1] M. Bernardo and R. Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202:1–54, 1998.

[2] B. Bloom and A.R. Meyer. A remark on bisimulation between probabilistic processes. In Meyers and Tsailin, editors, *Logik at Botik*. Springer-Verlag, 1989.

[3] D. Cazorla, F. Cuartero, V. Valero, and F.L. Pelayo. A process algebra for probabilistic and nondeterministic processes. *Information Processing Letters*, 2001. To appear.

[4] F. Cuartero, D. de Frutos, and V. Valero. PCSP: A denotational model for probabilistic processes. In *Proceedings of Third AMAST Workshop on Real-Time Systems*, 1996.

[5] F. Cuartero, D. de Frutos, and V. Valero. A sound and complete proof system for probabilistic processes. In *Proceedings of Fourth AMAST Workshop on Real-Time Systems*, 1997.

[6] P. R. D'Argenio, J. P. Katoen, and E. Brinksma. An algebraic approach to the specification of stochastic systems. *Programming Concepts and Methods*, pages 126–147, 1998.

[7] 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*, Sea of Galilee, Israel, 1990.

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

[9] H. Hermanns, U. Herzog, and J. P. Katoen. Process algebra for performance evaluation. *To appear in Theoretical Computer Science*, 1999.

[10] J. Hillston. A compositional approach to performance modelling. In *Cambridge University Press*. 1996.

[11] ISO/IEC 13818–2 Draft International Standard Generic Coding of Moving Pictures and Associated Audio. Recommendation H.262.

[12] B. Jonsson and K.G. Larsen. Specification and refinement of probabilistic processes. In *LICS'91*, Amsterdam, 1991.

[13] F. L. Pelayo, F. Cuartero, V. Valero, and D.Cazorla. PPNAL: Performance evaluation in an algebraic model for probabilistic an non-deterministic processes. In *Proceedings of 15th Annual UK Performance Engineering Workshop (UKPEW'99)*, pages 1–12, Bristol, U.K., 1999.

[14] F. L. Pelayo, F. Cuartero, V. Valero, and D.Cazorla. An example of performance evaluation by using the stochastic process algebra: ROSA. In *Proceedings of the 7th International Conference on Real-Time Computing Systems and Applications (RTCSA'2000)*, Cheju Island, Suth Korea, 2000.