Electronic Notes in Theoretical Computer Science 80 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume80.html 18 pages

Analysis of the MPEG-2 Encoding Algorithm with *ROSA* [1]

Fernando L. Pelayo [2] Fernando Cuartero [3] Valentín Valero [4]
Diego Cazorla [5]

Departamento de Informática
Universidad de Castilla-La Mancha
Escuela Politécnica Superior de Albacete. 02071 - SPAIN

Abstract

The authors present both the specification and a performance analysis of the MPEG–2 algorithm for video encoding, by using the Stochastic Process Algebra **ROSA**. This process algebra is a very general framework for describing and analyzing more complex Real Time Systems than the one presented. Some interesting results about the temporal behaviour of the algorithm and an immediate estimation of benefits when having a twin-processors platform have been obtained.

Keywords: Performance Analysis, Time Requirements, MPEG–2 Encoding Algorithm, Formal Methods, Stochastic Process Algebra.

1 Introduction

In the eighties, fundamentals in process algebras were firmly established, but the proliferation of distributed, real-time 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 algebras can be found in [3]. A pioneer work on a probabilistic extension of CCS is PCCS [9], where the non-deterministic choice is replaced by a probabilistic choice operator. Later, TPCCS is defined in [10], *Timed Probabilistic Calculus of Communicating Systems*, which extends CCS with probabilities and time.

[1] This work has been supported by the CICYT project "Performance Evaluation of Distributed Systems",TIC2000-0701-C02-02.
[2] Email:fpelayo@info-ab.uclm.es
[3] Email:fernando@info-ab.uclm.es
[4] Email:valentin@info-ab.uclm.es
[5] Email:dcazorla@info-ab.uclm.es

There are also some extensions of CSP [6,7], 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 all the possible behaviours of the system to model [14]. Unfortunately, it is not always possible to have such a knowledge.

Thus, part of the recent work of the authors 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; This language is presented in [4,5], where an operational, a denotational and a testing semantics have been defined for that language. The language used by the authors, **ROSA** (**R**easoning **O**n **S**tochastic **A**lgebras), has some functional similarities with PNAL.

However, 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 from the functional point of view, i.e., once it has been fully functionally tested. But, it becomes obvious that there are some systems (real-time systems) in which 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 considering these capabilities are presented in [12,8,11,2]. We can find in [2] the language EMPA *Extended Markovian Process Algebra*, which has two kinds of actions and the *race policy* is applied to decide the actions to be executed. However, a syntactic restriction is introduced in order to define the behaviour of synchronizations, because one of the actions involved in a synchronization must be always a passive one, which makes 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, since the EMPA process $(< a, \infty_{l,w} > . \underline{0} + < b, \infty_{l',w'} > . \underline{0}) \,||_{\{a\}}\, ||_{\{a\}} < b, * > . \underline{0}$, taking $l > l'$, cannot execute any actions at all.

ROSA [18] follows some ideas that the authors presented in [17]. 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.

ROSA is a Markovian Process Algebra which is able to capture non-determinism, probabilities and time aspects; in the case study here presented only has been exploited the last feature because the authors have no probabilistic information about the behaviour of the encoder yet, such as probability of components failure ... which could be captured by the probabilistic choice operator provided by this language.

ROSA has been used in order to analyze the performance of a real video encoding algorithm, the MPEG–2. This video encoder is the most extended version within the MPEG standards. The MPEG–2 video encoder was designed with two requirements in mind, namely, need for high compression, and need for random access capability. The first requirement is achieved by exploiting the spatial and temporal redundancy within an image sequence. The latter by considering a special kind of pictures (I pictures) which are coded with no reference to other frames, exploiting only spatial correlation in a frame.

In the literature we can found several works on performance improving of MPEG standards. Most of these works focus their improvements on parallelizing the distribution of data among the processors by either distributing different partitions of the same frame (spatial parallelism) or different GoPs to the various processors (temporal parallelism) [1,15,16].

Formal Methods (Process Algebras, Petri Nets, Markov Chains) try to analyze the potential improvement obtained when parallelizing algorithms, thus in [19] the authors presented a formal study of the MPEG–2 with Petri Nets. Indeed, a stochastic process algebra like **ROSA** allows to obtain some possible improvements by parallelizing either the encoding process of each B-frame within the GoPs or the encoding code itself.

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

2 Syntax of the Language

Let $\Delta = \{a, b, c, ..\}$ be a set of action types. We will denote by letters r, s, t probabilities, and by greek letters time parameters for actions.

Terms of **ROSA** are defined by:

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

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

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 which types belong to A) and $recX.P$ is the recursion operator.

The informal interpretation of the new operators follows:

- $\langle a, \lambda\rangle.P$ 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 $a.P$ is now 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.

- $P \oplus_r Q$ 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 style by using a labelled transition system with three kinds of transitions:

- Non-deterministic transitions:

$$P \longrightarrow Q$$

which represent the internal decisions that the system makes for resolving the non-deterministic choices. These evolutions do not take any time at all, see Table 1.

(ND-Def)	$\frac{}{P \oplus Q \longrightarrow P}$	**(ND-Def)**	$\frac{}{P \oplus Q \longrightarrow Q}$
(ND-Ext)	$\frac{P \longrightarrow P'}{P + Q \longrightarrow P' + Q}$	**(ND-Ext)**	$\frac{Q \longrightarrow Q'}{P + Q \longrightarrow P + Q'}$
(ND-Pro)	$\frac{P \longrightarrow P'}{P \oplus_r Q \longrightarrow P' \oplus_r Q}$	**(ND-Pro)**	$\frac{Q \longrightarrow Q'}{P \oplus_r Q \longrightarrow P \oplus_r Q'}$
(ND-Par)	$\frac{P \longrightarrow P'}{P\vert\vert_A Q \longrightarrow P'\vert\vert_A Q}$	**(ND-Par)**	$\frac{Q \longrightarrow Q'}{P\vert\vert_A Q \longrightarrow P\vert\vert_A Q'}$
(ND-Rec)	$\frac{}{recX.P \longrightarrow P[recX.P/X]}$		

Table 1
Non-Deterministic Transition Rules

- Probabilistic transitions:

$$P \longrightarrow_r Q$$

this rule represents that the process P can evolve with probability r to process Q.
  - **Action** is a boolean function defined on the set of **ROSA** processes which gives TRUE for those processes that can only evolve by means of *Action transitions* and FALSE otherwise.

These evolutions have not any temporal cost, see Table 2.

(P-Def)	$\dfrac{}{P \oplus_r Q \longrightarrow_r P}$	**(P-Def)**	$\dfrac{}{P \oplus_r Q \longrightarrow_{1-r} Q}$
(P-Ext)	$\dfrac{P \longrightarrow_r P' \wedge \textbf{Action}[Q]}{P + Q \longrightarrow_r P' + Q}$	**(P-Ext)**	$\dfrac{Q \longrightarrow_t Q' \wedge \textbf{Action}[P]}{P + Q \longrightarrow_t P + Q'}$
(P-Par)	$\dfrac{P \longrightarrow_r P' \wedge \textbf{Action}[Q]}{P\|\|_A Q \longrightarrow_r P'\|\|_A Q}$	**(P-Par)**	$\dfrac{Q \longrightarrow_t Q' \wedge \textbf{Action}[P]}{P\|\|_A Q \longrightarrow_t P\|\|_A Q'}$
(P-BothExt)	$\dfrac{P \longrightarrow_r P' \wedge Q \longrightarrow_t Q'}{P + Q \longrightarrow_{r\cdot t} P' + Q'}$	**(P-BothPar)**	$\dfrac{P \longrightarrow_r P' \wedge Q \longrightarrow_t Q'}{P\|\|_A Q \longrightarrow_{r\cdot t} P'\|\|_A Q'}$

Table 2
Probabilistic Transition Rules assuming $\textbf{DS}[P] \wedge \textbf{DS}[Q]$

- Action transitions:

$$P \xrightarrow{a,\lambda} Q$$

which represent that P can evolve by executing the action labelled by a, taking a time described by an Exponential random distribution with parameter λ, to process Q. Three auxiliary functions are needed:
· **Available** is a function defined on the set of **ROSA** processes giving for every process its multiset of available actions.
· **Type** is a function defined on the set of multisets of actions giving for every multiset, the set consisting of their action types.
· **DeterministicStability, DS** is a boolean function defined on the set of **ROSA** processes which gives TRUE for those processes that can not evolve by means of *Non-deterministic transitions* and FALSE otherwise.

(A-Def)	$\dfrac{}{a.P \xrightarrow{a,\infty} P}$	**(A-Def)**	$\dfrac{}{\langle a,\lambda\rangle.P \xrightarrow{a,\lambda} P}$
(A-Ext)	$\dfrac{P \xrightarrow{a,\lambda} P' \wedge a \notin \textbf{Type}[\textbf{Available}[Q]]}{P + Q \xrightarrow{a,\lambda} P'}$	**(A-Ext)**	$\dfrac{Q \xrightarrow{a,\lambda} Q' \wedge a \notin \textbf{Type}[\textbf{Available}[P]]}{P + Q \xrightarrow{a,\lambda} Q'}$
(A-Par)	$\dfrac{P \xrightarrow{a,\lambda} P' \wedge a \notin \textbf{Type}[\textbf{Available}[Q]] \cup A}{P\|\|_A Q \xrightarrow{a,\lambda} P'\|\|_A Q}$	**(A-Par)**	$\dfrac{Q \xrightarrow{a,\lambda} Q' \wedge a \notin \textbf{Type}[\textbf{Available}[P]] \cup A}{P\|\|_A Q \xrightarrow{a,\lambda} P\|\|_A Q'}$
(A-RaceExt)	$\dfrac{P \xrightarrow{a,\infty} P' \wedge Q \xrightarrow{a,\lambda} Q' \wedge \lambda \neq \infty}{P + Q \xrightarrow{a,\infty} P'}$	**(A-RaceExt)**	$\dfrac{P \xrightarrow{a,\lambda} P' \wedge Q \xrightarrow{a,\infty} Q' \wedge \lambda \neq \infty}{P + Q \xrightarrow{a,\infty} Q'}$
(A-RacePar)	$\dfrac{P \xrightarrow{a,\infty} P' \wedge Q \xrightarrow{a,\lambda} Q' \wedge \lambda \neq \infty \wedge a \notin A}{P\|\|_A Q \xrightarrow{a,\infty} P'\|\|_A Q}$	**(A-RacePar)**	$\dfrac{P \xrightarrow{a,\lambda} P' \wedge Q \xrightarrow{a,\infty} Q' \wedge \lambda \neq \infty \wedge a \notin A}{P\|\|_A Q \xrightarrow{a,\infty} P\|\|_A Q'}$
(A-RaceExtCoop)	$\dfrac{P \xrightarrow{a,\lambda_1} P' \wedge Q \xrightarrow{a,\lambda_2} Q' \wedge (\lambda_1 = \infty = \lambda_2 \vee \lambda_1 \neq \infty \neq \lambda_2)}{P + Q \xrightarrow{a,\lambda_1+\lambda_2} P' \oplus Q'}$	**(A-Syn)**	$\dfrac{P \xrightarrow{a,\lambda_1} P' \wedge Q \xrightarrow{a,\lambda_2} Q' \wedge a \in A}{P\|\|_A Q \xrightarrow{a,\textbf{min}[\{\lambda_1,\lambda_2\}]} P'\|\|_A Q'}$
(A-RaceParCoop)	$\dfrac{P \xrightarrow{a,\lambda_1} P' \wedge Q \xrightarrow{a,\lambda_2} Q' \wedge (\lambda_1 = \infty = \lambda_2 \vee \lambda_1 \neq \infty \neq \lambda_2) \wedge a \notin A}{P\|\|_A Q \xrightarrow{a,\lambda_1+\lambda_2} P'\|\|_A Q \oplus P\|\|_A Q'}$		

Table 3
Action Transition Rules assuming $\textbf{DS}[P] \wedge \textbf{DS}[Q]$

A detailed description of the operational semantics of **ROSA** can be found in [18]

4 MPEG–2 Digital Video Coding Standard

The ISO/IEC 13818–2 standard [13], 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 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, see Fig. 1. 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.*

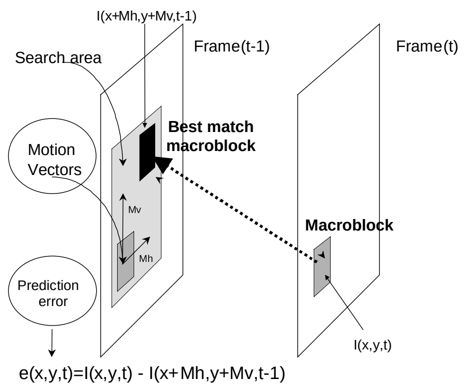


Fig. 1. Block diagram of the MPEG–2

In order to reduce spatial redundancy a coding method, DCT (Discrete Cosine Transform), is used. A major objective of this *transform domain coding* is to make small enough as many transform coefficients as possible, so that they are insignificant and need not to be coded for transmission. Low DCT coefficients are related to low spatial frequencies within image blocks and high DCT coefficients to higher frequencies. This property is used to remove subjective redundancies contained in the image data, taking into account the human visual systems criteria. Since the human viewer is more sensitive to reconstruction error related to low spatial frequencies than to high ones, 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.

The MPEG–2 has to achieve the requirement of random access and high compression, thus this standard specifies three types of compressed video frames/pictures: 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. They allow fast random access but offer moderate compression. P pictures (predictive coded pictures) are coded by using motion compensated prediction of a previous I or P picture. The compression for P pictures is higher than for I pictures. Finally, B pictures (bidirectionally–predictive coded pictures) are obtained by motion compensation from both past and future reference frames (I or P pictures), and provide the highest degree of compression.

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 pictures may be arranged in a sequence with a high degree of flexibility depending on the applications requirements. Thus, a video sequence coded using only I pictures allows the highest degree of random access, but achieves the lowest compression. A sequence code with I and P pictures (e.g. IPPPIPPP. . .) achieves moderate compression and certain degree of random access. Finally, a sequence that incorporates the three kinds of pictures (e.g. IBBPBBP. . .) may achieve high compression and reasonable random access, but also increases the coding delay significantly.

In order to understand how the MPEG–2 encoder works we will consider a GoP consisting of the frames IBBP, see Fig. 2 Despite B pictures appear before P pictures, the coding order is IPBB because B pictures require both past and future frames of the original video sequence as references.

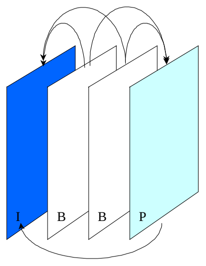


Fig. 2. Encoding order for an IBBP GoP

First, for every kind of frame, an elementary compression technique which makes use of specific physiological characteristics of human eye may be used: the human eye is more sensitive to changes in brightness than to changes in chromaticity. Therefore the MPEG–2 coding schema first divide images into YUV components (one luminance and two chrominance components). Then the chrominance components are subsampled relative to the luminance component with a ratio specific to particular applications (e.g. 4:1:1).

The first frame in a GoP (I picture) is encoded in intra mode without references 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 sent 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 compressed I picture which is stored temporarily in the Frames Store (FS), see Fig. 3.

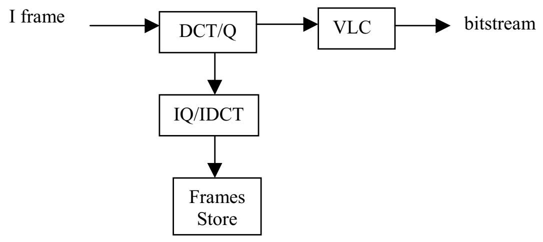


Fig. 3. Block diagram for I pictures

If the input is coded either as P or B pictures, 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 sent to the output buffer. As in the previous case, a compressed P picture is stored in the Frames Store, see Fig. 4.

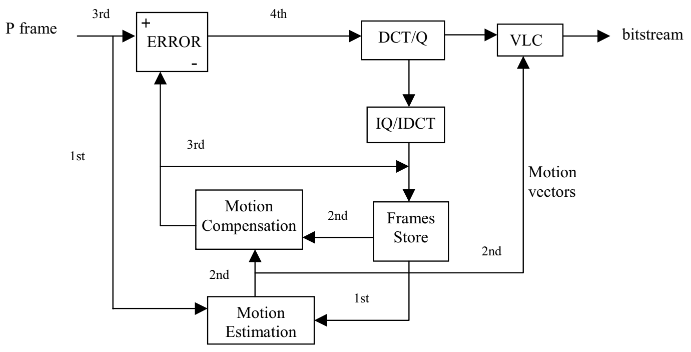


Fig. 4. Block diagram for P pictures

With B pictures, the motion estimation process is performed twice: 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 sent to the output buffer. It is not necessary to store in FS the compressed B picture since it will never be taken as reference, see Fig. 5.

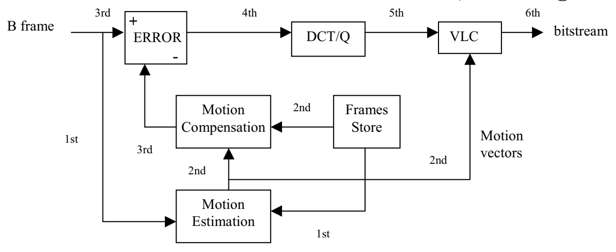


Fig. 5. Block diagram for P pictures

Finally, Fig. 6 includes all the information about the encoding process of every images from a GoP.

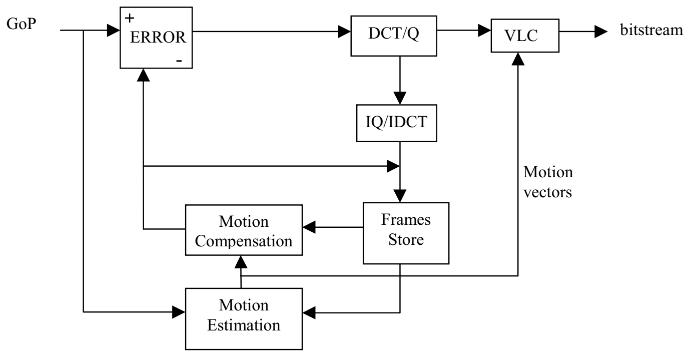


Fig. 6. Block diagram for P pictures

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 these steps:

(i) To begin with, non-deterministic transitions are eliminated from this *transition graph*, and we split the graph into several new graphs, according to Fig.7. Thus, we obtain a set of *deterministic transition graphs* which have only labelled transitions.

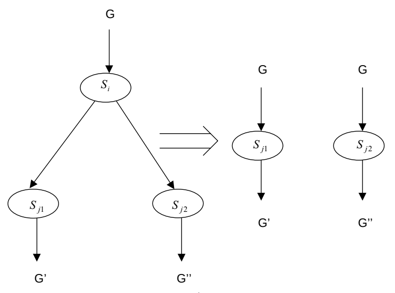


Fig. 7. Evaluation Algorithm - Phase 1

(ii) Now, from each of these *deterministic transition graphs*, we relabel every transition: $\xrightarrow{a,\lambda}$ with $\xrightarrow{1,1/\lambda}$, 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.8).

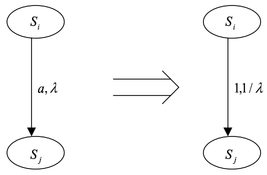


Fig. 8. Evaluation Algorithm - Phase 2

(iii) Next, for each graph, we remove both the probabilistic transitions and those with zero average duration by joining the nodes connected by those transitions in a single one as Fig.9 shows.

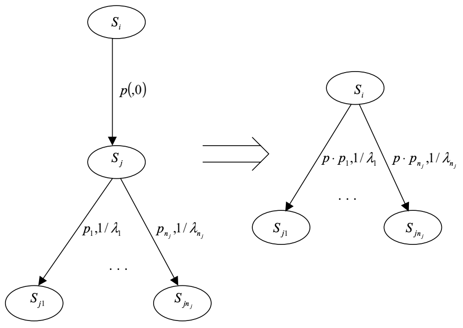
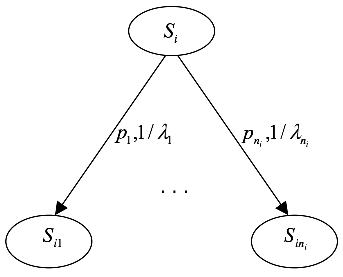


Fig. 9. Evaluation Algorithm - Phase 3

The probability of the outgoing edges are weighted by the probability of the eliminated one, since they can be considered as independent events and thus the probability can be multiplied.

Once we have carried out all these steps, we get a set of *deterministic reduced temporal graphs* which only contain arcs with non-zero temporal cost.

(iv) We have to identify the states/nodes from which we are interested in computing the necessary time in order to evolve from one called *initial state* and denoted by S_0, to another called *final state* and denoted by S_F, afterwards we have to prune the brunches which do not connect such states.



$$T_{S_i} = \sum_{j=1}^{n_i} p_j (1/\lambda_j + T_{S_{ij}})$$

Fig. 10. Evaluation Algorithm - Phase 4

We denote by T_{S_i} the expected time to evolve from the state S_i to the final state (S_F), thus we have $T_{S_i} = \sum_{j=1}^{n_i} p_j(1/\lambda_j + T_{S_{ij}})$, where n_i is 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 Fig. 10 shows.

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 times so 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 MPEG–2 Encoding Algorithm

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:

$B_i \equiv \langle EST_{B_i}, \beta_1\rangle.\langle COMP_{B_i}, \beta_2\rangle.\langle ERROR_{B_i}, \beta_3\rangle.\langle DCT\_Q, \beta_4\rangle.\langle VLC_{B_i}, \beta_5\rangle.out$

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

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

Table 4
MPEG-2 specification in **ROSA**

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, although the order of their encoding processes will be IPBB because of the particularities described in Section 4. Albeit the most common GoP is formed of the sequence IBBPBBPBBPBBPBBP, we have chosen the GoP IBBP since it has all types of images and the study of the industrial one is quite big, but similar in essence to the one here presented.

This specification [6] (table 4) lacks of any probabilistic behaviour. As

[6] Could be easier for the reader to follow in Fig. 3 the specification of the encoding process of frame I, and in Figs. 4 and 5 can be followed the specifications of the coding of P and B frames, respectively

soon as we have results concerning the probability of failure of any component/process this feature will be included in the analysis (in [18] can be seen an example of how the evaluation algorithm deals with probabilities).

Fig. 11 shows a partial view of 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 just encoded are marked with rectangular boxes.

We can observe in this figure 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.

Both evolutions can be seen separately in Fig. 12. Unfortunately these two evolution graphs have a pure sequential behaviour and will not show the evaluation algorithm managing with probabilities, as mentioned above.

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

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

If we compute the average times for encoding the I-image and the P-image, the values that we get for both graphs coincide:

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

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

However, when we compute the average time for encoding the B_1-image we obtain some different results.

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 video sequence in order to compare the theoretical results provided by **ROSA** with real measurements, specially those results concerning the range of time to encode the first B-frame.

The experiment was repeated several times. The results being reported have been condensed in three experiments which summarizes 3×10 trials. During the trials, no other operations were taken place in our experimental setup.

We have made the experiments with a video sequence representative of different levels of motion, the “Composed” sequence (format PAL CCIR–601, 720×576 pixels). We have used a completely software-based MPEG–2 video encoder derived from that developed in Berkeley which is freely available in *ftp://mm-ftp.cs.berkley.edu/pub/multimedia/mpeg/encode*, other versions could be obtained from the MPEG Home Page, *http://www.mpeg.org*.

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 the required 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:

$$\begin{array}{lll}
\alpha_1 = 1.681 & \alpha_2 = 3.089 & \alpha_3 = 1.486 \\
\gamma_1 = 0.847 & \gamma_2 = 31.658 & \gamma_4 = 1.920 \\
\gamma_5 = 2.600 & \gamma_6 = 1.942 & \beta_1 = 0.415 \\
\beta_2 = 20.626 & \beta_4 = 1.920 & \beta_5 = 2.613
\end{array}$$

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 the measures obtained in real codification experiments within the described scenario.

Mainly, we are interested in estimate how much it represents the difference between the slowest and the quickest traces to be encoded the first B picture of a GoP and what does it represent with respect to the time for encoding the whole GoP, specially the relation between the quickest trace and the GoP trace.

	ROSA	**Experiment 1**	**Experiment 2**	**Experiment 3**
I-Image	0.918613s	0.909285s	0.953689s	0.893104s
P-Image	3.70923s	3.700998s	3.738053s	3.688503s
B_1**-Image**	7.59s - 10.56s	10.020147s	7.813504s	9.988154s
GoP	10.9475s	11.549177s	10.793488s	11.067721s

Consequently, according to our **ROSA** specification, we can obtain the following bounds to the time necessary for encoding the first B-image: [7.58582s - 10.5648s].

But from the generated graphs it is immediate to conclude that the codification process of both B images can be done at once, so, provided that we use a platform with two processors the necessary time for encoding a GoP like the considered, can be reduced from 10.9s to 7.6s, i.e. about a 30%, with the only task for one of the processors that, once both I and P frames are in the Frames Store ready to be taken as reference for the encoding process of the next pair of B pictures, it just has to encode one of the B frames. In fact, the processor which only has to encode one of the B pictures could also be used previously for the task VLC of either I or P pictures for instance, so the benefit above estimated is a lower bound for the benefit expected when having a twin-processors platform.

Moreover, since we have used a very simple model of GoP in order to develop our study and the main advantages came from the simultaneous encoding of both B-images; we claim that having a more realistic GoP like IBBPBBPBBPBBPBBP, we could obtain with a twin-processor platform an improvement close to a 36%.

It would be very interesting to develop a more complete study of this algorithm in which the maximum parallelism in the encoding process of a GoP was considered, and then it can be analyzed both the necessary number of processors and the profit obtained in such a case. But, Classical Process Algebras lack of the necessary description capabilities, probably *Timed Arc Petri Nets* provide a suitable environment in order to do that.

7 Conclusions and Future Work

We have presented an application of Formal Methods to the primary performance analysis of a real algorithm for video compressing, the MPEG–2.

Specifically, a study of some performance features of this video encoding algorithm which consists of the estimation of the bounds for the time to encode each type of image and the minimum benefits obtained when having a twin-processor platform, has shown.

The authors' current work is mainly focused on:

- Analyzing another version of this algorithm taking into account the possibility of system failures. This would allow to show the benefits of having a probabilistic choice operator in **ROSA**.
- Analyzing and comparing two different parallel versions of the MPEG–2 encoding algorithm: Spatial Parallelism vs. Temporal Parallelism.
- Analyzing the potential degree of parallelism of the algorithm, in order to estimate the best performance which can be obtained under this assumption.

References

[1] S. M. Akramullah, I. Ahmad, and M. L. Lion. Performance of software-based MPEG-2 video encoder on parallel and distributed systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(4):687–695, 1997.

[2] 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.

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

[4] D. Cazorla, F. Cuartero, V. Valero, and F.L. Pelayo. A Process Algebra for Probabilistic and Nondeterministic Processes. *Information Processing Letters*, 80(1):15–23, October 2001.

[5] D. Cazorla, F. Cuartero, V. Valero, F.L. Pelayo, and J.J. Pardo. Algebraic Theory of Probabilistic and Nondeterministic Processes. *Journal of Logic and Algebraic Programming*, 55(1–2):57–103, March-April 2003.

[6] 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.

[7] 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 ARTS'97, LNCS 1231*, 1997.

[8] 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.

[9] 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.

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

[11] H. Hermanns, U. Herzog, and J-P. Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 274(1–2):43–86, 2002.

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

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

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

[15] T. Olivares, F. Quiles, P. Cuenca, L. Orozco, and I. Ahmad. Study of data distribution techniques for the implementation of an MPEG-2 video encoder. In *PDCS´ 99, Vol. 1*, pages 537–542, 1999.

[16] T. Olivares, F. Quiles, A. Garrido, P. Garcia, and L. Orozco-Barbosa. The need of multicast predictive nfs servers for high-speed networks used as parallel multimedia platforms. In *ICPP´ 01, IEEE Computer Society Press*, pages 391–396, 2001.

[17] 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.

[18] 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 IEEE International Conference on Real-Time Computing Systems and Applications (RTCSA'2000). IEEE Computer Society Press*, pages 271–278, Cheju Island, South Korea, 2000.

[19] V. Valero, F. L. Pelayo, F. Cuartero, and D. Cazorla. Specification and analysis of the MPEG-2 video encoder with Timed-Arc Petri Nets. *Electronic Notes on Theoretical Computer Science*, 66(2):125–136, *http://www.elsevier.nl/locate/entcs/volume66.html*.

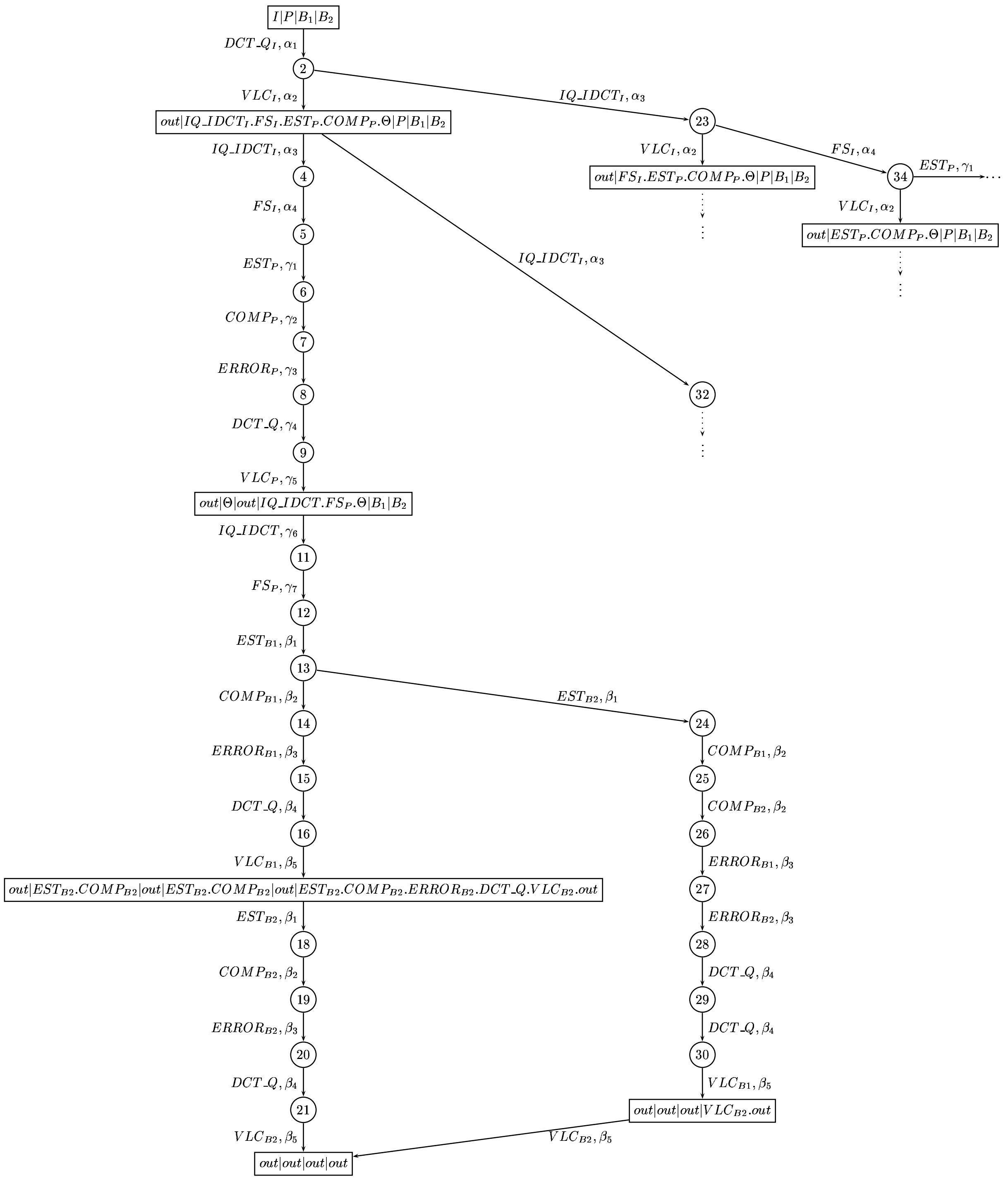


Fig. 11. *Deterministic Transition Graph of MPEG–2*

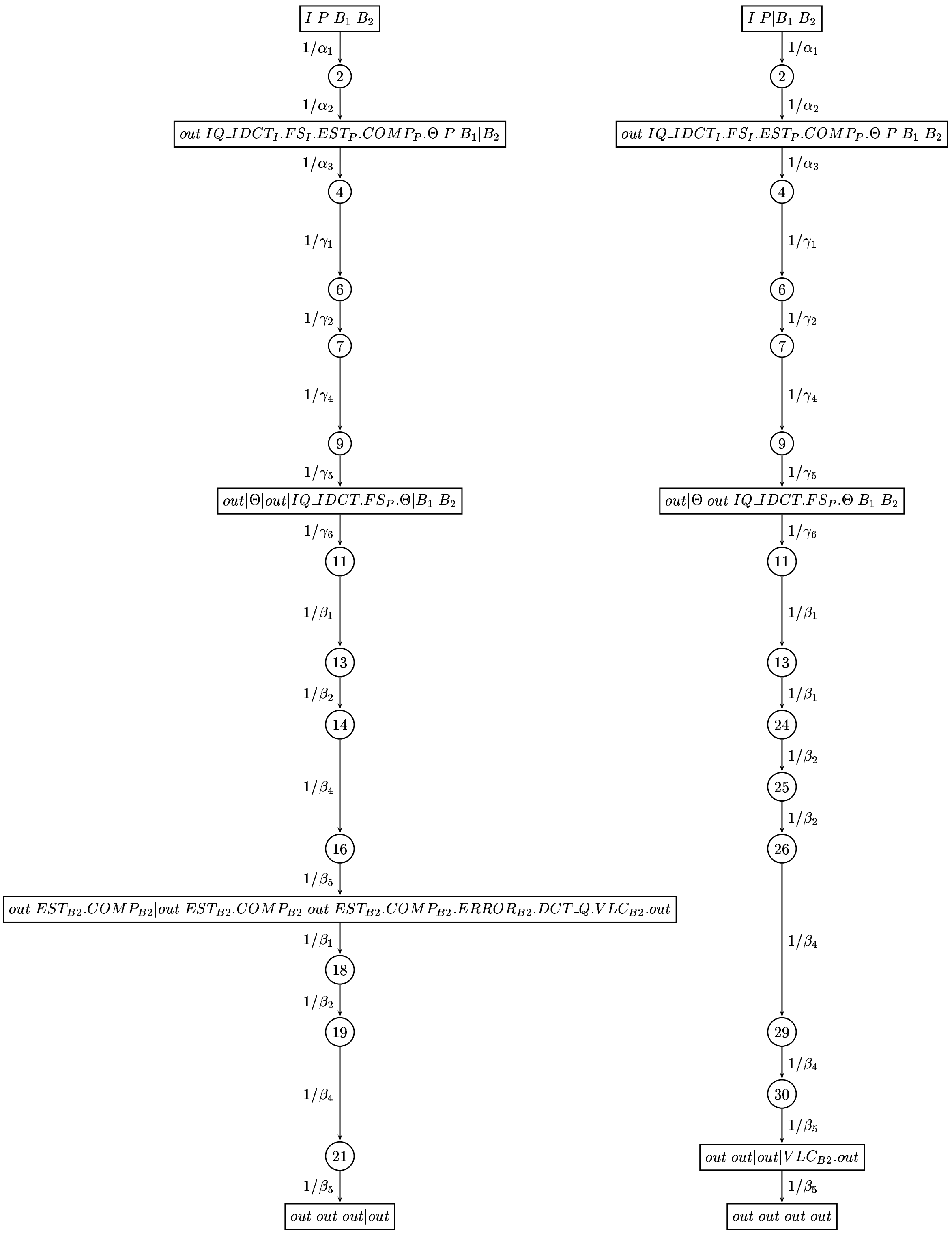


Fig. 12. *The Quickest and the Slowest encoding traces*