A translation of $TPAL_p$ into a class of timed-probabilistic Petri nets

Antonio Bueno*, Valentín Valero, Fernando Cuartero

Escuela Politécnica Superior de Albacete, Univ. de Castilla-La Mancha,
Campus Univ. s/n. 02071. Albacete, Spain

Received 26 November 2002; received in revised form 24 February 2005; accepted 25 February 2005

Communicated by G. Rozenberg

Abstract

$TPAL_p$ is an algebraic language for the description of concurrent systems with capabilities to express timed and probabilistic behaviours, as well as urgent interactions. In this paper we present the main features of the language, its operational semantics, and a translation of $TPAL_p$ terms into a particular class of timed-probabilistic Petri nets. The language includes a probabilistic choice operator, a timed prefix operator, and an urgent prefix operator, as well as some other operators that we may find in classical process algebras. An important feature of the language is that urgency is considered at any instant by executing as many urgent actions as possible, with the goal of complying in a great extent with the urgent actions indicated in the user specifications.

Keywords: Formal methods; Software specification; Concurrent systems; Process algebra; Timed Petri nets

1. Introduction

Two models are widely used for the modelling and analysis of concurrent systems, process algebras and Petri nets. Process algebras are structured description languages based on a few simple constructs (e.g., sequential and parallel composition, non-deterministic choice and recursion), which are capable to describe the various aspects of concurrent systems. Petri
nets have an important advantage with respect to process algebras, they have a graphical nature, and thus they are easier to interpret. Furthermore, there is a solid mathematical foundation supporting them, and some techniques for the systematic analysis of properties are firmly established. Thus, some years ago it became evident that a relationship between both formalisms would be very useful, because by doing so we could exploit the advantages of both description techniques. Goltz [17] presents a translation of CCS into Petri nets, Olderog [26] defines a translation of a language inspired on COSY, CCS and CSP into 1-safe Petri nets, while Taubner [31] defines such a translation for a more general process algebra, and more recently, a different approach has been introduced, the Petri Box Calculus [5], which takes the standard operators that we find in process algebras, applying them to a domain of Petri nets (Boxes). One of the main advantages of this latter approach is that we get a compositional behaviour of the resulting nets.

Nowadays we can find a wide range of systems for which time and probabilities become two important factors to be considered in the specifications (real-time systems and fault-tolerant systems). We also find in these systems that some actions or tasks are critical, in the sense that they must be executed urgently once they become enabled. Then, many timed and/or probabilistic extensions of both process algebras and Petri nets have been proposed. With regard to timed extensions of the classical process algebras we find in the literature many different proposals, we just mention some of them: timed CCS [35], temporal CCS [24], timed CSP [29], \textit{TPAL}_p [10], and timed ACP [3]. In the case of Petri nets there are essentially two trends when extending them with time: either time is associated with transitions (transition durations), or time is associated with places or tokens (representing delays for the corresponding tokens to be available). There is also a third group of models that include some kind of time information in the arcs, but in this case time must be also associated either with places, tokens or transitions. A survey of the different approaches to introduce time in Petri nets is presented in [8].

Concerning with probabilistic extensions of these models, we can cite [22] as a pioneer on this subject in the area of process algebras. Afterwards we can find probabilistic versions of CCS [16], CSP [13,14,25], and there are also some probabilistic extensions of these models that maintain non-determinism [11,36].

There are also some previous works that join in a single model time and probabilities. In this line Hansson and Jonsson have defined a timed and probabilistic extension of CCS [21,20], and in [12] a specification language that incorporates both probabilistic and timing aspects of process behaviour is defined (PDP). In [18] a testing semantics for an extension of LOTOS with time and probabilities is also presented. More recently, [2] presents an ACP-like process algebra to model both probabilistic and timed behaviours, and [9] defines a variant of CSP that considers probabilistic external/internal choices, multiway synchronisations and the possibility to model processes with different advancing speeds.

\textit{TPAL}_p (Timed-Probabilistic Alternating Language) [10] is an algebraic language for the description of concurrent systems which also integrates both factors: time and probabilities. The time model we consider in \textit{TPAL}_p is discrete, there is a special action \(i\) that represents the passage of one unit of time, as the tick action in [21]. The alternative to discrete time models are continuous time models, which are normally preferred, as they use a real time scale, and thus, the execution of actions is not restricted to discrete points in time. However, as Baeten and Middelburg mention in [4] measuring time on a discrete time scale does not
mean this, but to divide time into slices and timing of actions is done with respect to the time slices in which they are performed. Actually, computers measure time by means of discrete clocks, and if they are used to control a physical system, the state of the physical system is sampled and adjusted at discrete points in time.

The main features of TPAL\textsubscript{p} are the following: there is a probabilistic choice operator in the line of the generative probabilistic extensions of process algebras [16], specifying a probability distribution for the components. With this interpretation, the system decides which component is going to be executed according to the indicated probability distribution. Furthermore, following the ideas of Quemada et al. [28], TPAL\textsubscript{p} includes a timed prefix operator, which establishes a relative time interval for the execution of the corresponding action (relative with respect to the time at which the preceding action is performed), and once the deadline for the action has expired this action cannot be executed ever after. Therefore, the timing of actions is relative in TPAL\textsubscript{p}, as in nearly all process algebras with timing. The alternative to relative timing is absolute timing, but traditionally it has been discarded, as it is considered more difficult than relative timing. Nevertheless, Baeten and Middelburg have shown in a recent work [4] that process algebras with absolute timing are not really more difficult to use in describing and analysing the time-dependent behaviour of systems.

A third operator included in TPAL\textsubscript{p} is the urgent action prefix, which establishes the urgent character of the involved action, in the sense that it must be executed once the environment requires it, without any delay (unless we have a conflict with other urgent actions).

The usual operators that we may find in classical process algebras are also included in TPAL\textsubscript{p}: external choice, parallel operator, hiding and recursion. Following the same ideas of [36], in TPAL\textsubscript{p} we have probabilistic processes and non-deterministic processes, and the syntax of the model imposes the alternation of both probabilistic and non-deterministic terms, in a similar way to that followed in [21]. Therefore, probabilistic choices and external choices are strictly separated, which makes easier the formal study, as Hansson and Jonsson mention in their work [21]. There are some recent works addressed to the definition of probabilistic models considering both kinds of choice without these syntactical restrictions [11,13,14,25]. However, it is noteworthy that all these works show that each syntactical term can be represented in an equivalent way by another one (its normal form) which follows an alternating syntax.

A consequence of this alternating syntax is that the operational semantics consists of two kinds of transition rules (probabilistic vs. non-deterministic), which is not surprising, because our probabilistic transition rules are in some way equivalent to the internal transitions of other models. This operational semantics treats the urgent actions specially, considering that they are in some way more important, and thus, it tries to execute as many of them as it can.

In this paper we also present a translation of a particular class of TPAL\textsubscript{p} terms into a model of timed-probabilistic Petri nets. The terms for which this translation works are those closed and guarded terms for which no free identifier appears in a subterm affected by either a parallel or a hiding operator (following the terminology of [26] they will be called regular terms). With these restrictions the obtained nets will be finite, which is an important property of a workable net semantics.

The model of timed-probabilistic Petri nets we use are Timed-Arc Petri nets [1,7,15,19,32,34] extended with probabilities (TPNets). In these nets tokens have
associated a non-negative integer value indicating the elapsed time from its creation (its age), and arcs from places to transitions are also labelled by time intervals, which establish restrictions on the age of the tokens that can be used to fire the adjacent transitions. As a consequence of these restrictions some tokens may become dead, because they will never be available, since they are too old to fire any transitions in the future.

The probabilistic extension of Timed-Arc Petri nets that we consider in this paper is based on the introduction of a new class of places, the so-called decision places, whose outgoing edges are labelled with a probability. When some of these places are marked, the system must make a probabilistic decision, and no time can elapse until reaching a marking with no decision place marked.

The paper is structured as follows: in Section 2 we present an intuitive overview of our approach. In Section 3 we present the basic notations, the syntax of TPAL, and the definition of the feasible bags of a process. The operational semantics of the language is defined in Section 4. In Section 5 we present some important definitions for the mapping of terms in nets. Then, the Net model is defined in Section 6, and the Net semantics in Section 7, as well as the transfer lemmas that ensure the correctness of this semantics.

2. Overview

This work is inspired by the studies of Olderog in [26] about the behaviour of concurrent processes described by Petri nets, algebraic terms and logical formulas. In this work a link between these different description methods is presented, in particular, an operational Net semantics that assigns an abstract net to every process term, and it is proved that this net admits exactly the same computations as those obtained from the operational semantics of the algebraic model. Furthermore, this translation fulfils the so-called Concurrency Principle, which requires that modulo strong bisimilarity on nets, the operational net semantics must be compositional with respect to the standard compositional net operators.

Then, our goal in this paper is to extend that work to TPAL, i.e., we define a link between the terms of TPAL and the timed-probabilistic Petri nets we consider (TPNets). We now present a brief overview of the algebraic model and the Net model, and we use some examples to illustrate how the translation works.

2.1. The algebraic model

As we have mentioned in the introduction, terms of TPAL have an alternating syntax, and there is a special action i that appears in front of every probabilistic subterm. For instance, i; stop and i; a; i; stop +0.4 i; stop are probabilistic terms. In the first case the only possible evolution (with probability 1) is that one unit of time elapses, and then it deadlocks (time passes, but no action is executed). In the second case, the number that labels the operator (+) indicates the probability of each alternative, in this case with probability 0.4 it can evolve to a; i; stop, and with probability 0.6 to i; stop, and one unit of time passes in both cases. The action a in a; i; stop is urgent, as it does not have associated a time interval, this means that no time can pass and it must be executed immediately. However, if we now consider the term a(1, 4); i; stop, the action a can be executed at any instant in the relative time interval
[1, 4], but we are not forced to execute the action even if the time interval is going to expire.

The external choice operator is \( \square \) and the parallel operator \( \parallel_A \), where \( A \) is the set of synchronisation actions. Both basically have a classical interpretation, but taking into account that at every moment we are forced to execute a maximal number of urgent actions. This requirement can resolve the choices in some cases, for instance, the process \( N = a; i; \text{stop} \parallel b(0, 0); i; \text{stop} \) can only evolve by executing the action \( a \), because it is urgent. Nevertheless, we cannot forget the second component of this choice merrily, because \( b \) could become urgent if this term is part of a parallel process, as it occurs in \( N \parallel \{b\} b; i; \text{stop} \), where the urgent character of \( b \) in the second argument is inherited by the action \( b \) that \( N \parallel \{b\} b; i; \text{stop} \) can execute as result of the synchronisation.

Hiding is denoted by the backslash operator \( \backslash \), thus, \( a(0, 5); i; \text{stop}\backslash a \) is a process that behaves as \( a(0, 5); i; \text{stop} \), but executing an internal action \( \tau \) instead of \( a \). The last operator of \( TPAL_p \) is the recursion, which allows us to describe infinite behaviours. For instance, \( \mu X.a(0, 1); i; X \) is a process that can execute an infinite sequence of actions \( a \), but one or two ticks must always elapse between two consecutive ones.

2.2. The net model

\( TPNets \) are a probabilistic extension of Timed-Arc Petri Nets [15]. Places of \( TPNets \) can be either normal places or decision places. Normal places are similar to the places of Timed-Arc Petri Nets, the outgoing arcs from these places must be labelled either with a time interval or with ‘∗’. Instead, arcs leaving from decision places are labelled with probabilities, and the addition of all these probabilities must be 1.0. Graphically, normal places are pictured by a circle, as usual, and decision places will be pictured by a double circle, in order to distinguish them easily. \( TPNets \) have also two types of transitions, normal and urgent. They are represented by a box, as usual, but urgent transitions are distinguished filling them in black.

\( TPNets \) must also fulfil the following restrictions:
- \( TPNets \) are T-restrictive (every transition has at least one precondition place and one postcondition place).
- Arcs connecting normal places with normal transitions are labelled by a time interval.
- Arcs connecting normal places with urgent transitions are labelled either with a time interval or with ‘∗’.
- Transitions are labelled with actions, but postcondition transitions of decision places must be labelled with the internal action \( \tau \).
- Postcondition transitions of decision places have only one precondition place (the decision place).

An example of \( TPNet \) is shown in Fig. 1, where \( \{p1, \ldots, p8\} \) are normal places, \( \{d1\} \) is a decision place, \( \{t1, t2, t3\} \) are normal transitions, and \( \{u1, u2, u3\} \) are urgent transitions.

Tokens are annotated with an age, as it occurs in Timed-Arc Petri Nets, and they are graphically represented by their age (\( p1 \) and \( p2 \) in Fig. 1 have both one token of age 0). Then, the age of a token is growing until the token is consumed. However, a consequence of the urgent character of the postcondition transitions of decision places is that tokens on these decision places can never grow, i.e., their age must be always 0, because
these tokens must be used immediately to fire one of the postcondition transitions of these places.

Time intervals on the arcs limit the age of the tokens that can be used to fire the adjacent transition. For example, \( t_1 \) in Fig. 1 can only be fired if we have a token \( x \) in \( p_1 \) with an age in the time interval \([1, 2]\), and a token \( y \) in \( p_2 \) with an age in the time interval \([0, 3]\). Therefore, in this particular case, \( t_1 \) can be fired at instants 1 and 2, but as it is not urgent we are not forced to fire it, so we can reach a marking in which both tokens have age 3, and both are dead, because they cannot be used to fire any other transition in the future.

Arcs labelled with ‘∗’ always point to urgent transitions. They are used to indicate that any token on the adjacent place can be used (whichever age it has) for the firing of the adjacent transition, but, of course, the urgent character of the transition is maintained.

The semantics of the Net model imposes that probabilistic decisions must be first resolved, without consuming any time. Then, once no decision place is marked the net can evolve by executing a multiset of transitions, but a maximal number of urgent transitions must be fired at any instant. With the firing of a multiset of transitions (which can be empty) one tick elapses, but notice that time can only elapse without firing any transitions when no urgent transitions are enabled. In the example of Fig. 1 the firing of \( t_1 \) leads to a marking in which \( d_1 \) and \( p_3 \) are both marked with one token of age 0. At this state, since \( d_1 \) is a decision place, we must first resolve the probabilistic decision, so we can execute \( u_1 \) with probability 0.3 and \( u_2 \) with probability 0.7. In case \( u_1 \) has been executed \( p_4 \) becomes marked with one token of age 0, and \( p_3 \) remains marked with one token of age 0, since no time has elapsed. Then, \( u_3 \) must be fired, since it is urgent, and with this firing one tick elapses, so the new marking consists of one token with age 1 in \( p_4 \) and one token of age 0 in \( p_6 \). Finally, this net can only fire transition \( t_2 \), and this can be made at the current instant, or in the following two instants, but notice that we are not forced to fire it, because it is not urgent.

2.3. Translation

The translation of a regular term of \( TPAL_p \) is made by splitting the term into sequential components, as in [26], but taking into account that we also have another class of terms (probabilistic). Sequential terms are easily translated, because they just consist of an only sequential component (the term itself). For instance, the \( TPNet \) associated with the term \( a; i; b(1, 3); i; \) stop is shown in Fig. 2.
External choices are also easily translated. The term \(a \langle 1, 3 \rangle; i; \text{stop} \parallel \{a, b\}\langle 1, 1 \rangle; i; c \langle 3, 3 \rangle; i; \text{stop} \) has again an only sequential component, and its corresponding TPNet is shown in Fig. 5.

Probabilistic choices are translated by means of decision places. For instance, the term \(a \langle 0, \infty \rangle; (i; b; i; \text{stop} +_p i; c \langle 3, 3 \rangle; i; \text{stop})\) only consists of a sequential component, and its corresponding TPNet is shown in Fig. 4.

Let us now consider the term \(a \langle 0, \infty \rangle; (i; b; i; \text{stop} \parallel \{a, b\}\langle 1, 1 \rangle; i; c; i; b \langle 0, \infty \rangle; i; \text{stop})\). Then, its corresponding TPNet is generated by putting together the TPNet of both components, but joining the transitions with labels in the synchronisation set \(\{a, b\}\). The resulting net is shown in Fig. 5.

Hiding is easy to translate, we just need to change the labels of the hidden actions, by replacing them with \(\tau\). For example, the TPNet of the term \(a \langle 0, 0 \rangle; (i; (b \langle 1, 1 \rangle; i; \text{stop}) \backslash b \parallel (c; i; \text{stop}) \backslash c)\) is shown in Fig. 6.

The translation of a recursive term \(\mu X. P\) is a bit more involved. Initially, the term is unfolded by replacing every occurrence of \(X\) in \(P\) by \(\mu X. P\), which is denoted by \(P \{\mu X. P / X\}\), and then this term is translated as usual, but taking into account that every occurrence of \(X\) is translated by connecting its precondition transition with the initial place of \(P\). Notice that every occurrence of \(X\) must be guarded, so it has an only precondition in the net, and
regularity forbids the parallel operator inside a recursion, so $P$ only can have one initial place (Fig. 7).

3. Definitions

3.1. Notation

As usual, $\mathbb{N}$ will represent the set of natural numbers, $\mathbb{N} = \{0, 1, \ldots\}$. The notation for working on multisets will be as follows: given a set $X$, the set of multisets over $X$ will be denoted by $\mathcal{B}(X)$, and given a multiset $B$ over $X$ we will indicate the number of instances of each element of $X$ in $B$ in the following way: $B = \{2.x_1, 3.x_2, 5.x_3\}$. 
Capitals $B$, $B_1$, $B_2$, $U$, $U_1$, $U_2$ will represent multisets, while $A$, $A_1$, $A_2$ will represent sets. The empty multiset will be also represented by $\emptyset$, because the context will make it evident if it is a set or a multiset.

We will use the following operations on multisets:

- $|B|$: Number of elements of $X$ in $B$, i.e., $|B| = |\{x \in X / B(x) > 0\}|$
- $:\text{Card}(B)$: This is the total number of elements in $B$, i.e.,
  \[ \text{Card}(B) = \sum_{x \in X} B(x) \]
- $\sharp(x, B)$:Multiplicity of $x$ in $B$, i.e., $\sharp(x, B) = B(x)$.
- $\text{Enum}: B(X) \rightarrow \mathcal{P}(\mathbb{N} \times X)$: This is a function which enumerates the elements of a multiset, in the following way:
  \[ \text{Enum}(\{2, x_1, 3, x_2\}) = \{(1, x_1), (2, x_1), (1, x_2), (2, x_2), (3, x_2)\} \]
- $B_1 + B_2$: Union of multisets, i.e., $(B_1 + B_2)(x) = B_1(x) + B_2(x)$ for all $x$.
- $B_1 \cap B_2$: Intersection of multisets, $(B_1 \cap B_2)(x) = \min\{B_1(x), B_2(x)\}$.
- $B_1 \subseteq B_2$: Inclusion for multisets, $B_1 \subseteq B_2$ if $B_1(x) \leq B_2(x)$ for all $x$.
- $B \setminus A$: Restriction of a multiset: this is the multiset obtained by taking out from $B$ all the actions belonging to the set $A$, i.e., $(B \setminus A)(x) = 0$ if $x \in A \subseteq X$, and $(B \setminus A)(x) = B(x)$, if $x \notin A$.
- $B \downarrow A$: Restriction over a set: this multiset consists of all the actions in $B$ and $A$, i.e., $(B \downarrow A)(x) = 0$, if $x \notin A$, and $(B \downarrow A)(x) = B(x)$, if $x \in A$.
- $B_1 +_A B_2$: Synchronisation of multisets on a set: this is the multiset obtained by adding the number of instances of each action not belonging to $A$ in both multisets, and taking the intersection for those in $A$, i.e., $(B_1 +_A B_2)(x) = B_1(x) + B_2(x)$, if $x \notin A$, and $(B_1 +_A B_2)(x) = \min\{B_1(x), B_2(x)\}$, if $x \in A$.
- $B_1 \ll_A B_2$: Urgency expansion: this is the multiset obtained when we take the intersection of both multisets for the actions in $A$, and the actions of $B_1$ not belonging to $A$, i.e., $(B_1 \ll_A B_2)(x) = B_1(x)$, if $x \notin A$, and $(B_1 \ll_A B_2)(x) = \min\{B_1(x), B_2(x)\}$, if $x \in A$.

We will also use the following operators:

Let $\bar{C}$, $\tilde{C}_1$, $\tilde{C}_2 \in \mathcal{P}(B(X) \times B(X))$:
- $\bar{C} \setminus A = \bigcup_{(B, U) \in \bar{C}} (B \setminus A, U \setminus A)$
- $\bar{C} \oplus_A \tilde{C}_2 = (\bar{C} \setminus A) \cup (\tilde{C}_2 \setminus A) \cup \bigcup_{(B_1, U_1) \in \bar{C}_1, (B_2, U_2) \in \tilde{C}_2} (B_1 +_A B_2, (U_1 \ll_A B_2 + U_2 \ll_A B_1) \ll_A (B_1 +_A B_2))$

This operator $\oplus_A$ will be useful when defining the semantics for the parallel operator.

### 3.2. Syntax of $TPAL_p$

Let $Act$ be a finite set of actions. We will use a special action $i$ ($i \notin A$) to represent the passage of one unit of time.

The syntax of $TPAL_p$ is defined as follows:

$$TPAL_p ::= N | P$$
where $N$ and $P$ are defined by the following BNF-expressions:

\[
N ::= \text{stop} \mid a(t_1, t_2); P \mid N \sqcup N \mid N \parallel_A N \mid a; P \mid N \setminus a \mid X \mid \mu X. N
\]

\[
P ::= P +_q P \mid i; N \mid P \parallel_A P \mid P \setminus a
\]

where $a \in Act$, $A \subseteq Act$, $0 \leq t_1 \leq t_2$, with $t_1 \in \mathbb{N}$, $t_2 \in \mathbb{N} \cup \{\infty\}$, $X \in Idf$ (set of identifiers) and $q \in [0, 1]$.

$N$ defines the behaviour of non-deterministic processes, while $P$ defines probabilistic ones. Then, $N_1$, $N_2$, ..., will represent processes defined by $N$, and $P_1$, $P_2$, ... will be processes defined by $P$. Capitals $X$, $Y$, ... will represent identifiers. The alternation of both kinds of terms becomes now apparent from the syntax, and we see that the action $i$ plays an important syntactic role, because we can observe that probabilistic terms have all their components prefixed by $i$.

Let us now briefly describe the informal meaning of the introduced operators (details will become clear when defining the operational semantics).

- **stop**: This represents a deadlock, i.e., no action, except $i$, can be executed.
- **$a(t_1, t_2); P$**: The action $a$ can be executed at any instant in the time interval $[t_1, t_2]$ (these times being relative to the current instant). Once the deadline has expired, this action cannot be executed ever after. However, there is no obligation for executing the action, even if the deadline is about to expire. As usual, once the action has been executed, this process behaves as the probabilistic term $P$.
- **$N_1 \sqcup N_2$**: This is essentially the classical external choice operator, i.e., in the absence of urgent actions, when a bag is requested to be executed, it will be executed by the non-deterministic process that can execute it, and as usual, when both processes are able to execute this bag, this choice is in fact internal (but not probabilistic), i.e., the system chooses internally the process which executes the bag, but this selection does not use any probabilistic information at all. When urgent actions are involved, the only enabled bags are those having the maximum number of urgent actions.
- **$N_1 \parallel_A N_2$**: This represents the parallel execution of $N_1$ and $N_2$, synchronising on the actions in $A$. Bags of $N_1$ and $N_2$ are then combined adding the instances of actions not belonging to $A$, and taking the intersection for those in $A$. Furthermore, those actions which are urgent on one side, become now urgent for the complete process. This extension of the urgent character of the actions involved on each side makes it a bit tedious the definition of the semantics for this operator. We need this operator for probabilistic processes too, in order to define their semantics properly.

We will denote by $\|\|$ the interleaving operator, i.e., $N_1 \| N_2 = N_1 \|_0 N_2$.

- **$a; P$**: This is the **urgent interaction operator**, which establishes that $a$ is urgent, and it must be executed without any delay, if the environment requires it. Once this action has been executed, the process behaves as $P$. Notice that it would be interesting to have another operator related with urgency, $a(n); P$, stating that $a$ will become enabled (and urgent) only after $n$ units of time. However, this operator can be derived, according to the expression: $a(n); P = a(n, n); P \parallel_1 a; i; \text{stop}$
- **$N \setminus a$**: This process behaves like $N$, but hiding the action $a$ (its executions are replaced by the internal action $\tau$). This is just a replacement, i.e., it does not affect the possible urgent character of the action $a$ in $N$. We also need this operator for probabilistic processes, just like as for the parallel composition.
• \( \mu X.N \): The recursion operator, which allows us to define infinite behaviours. As usual, we impose the restriction for these terms to be guarded (all the appearances of \( X \) in the components of \( N \) must be prefixed).

• \( P_1 +_q P_2 \): This represents the probabilistic choice between \( P_1 \) and \( P_2 \), according to the specified probability \( q \). This choice is fully probabilistic, i.e., it does not matter at all the environment requirements. Notice that the imposed alternation between probabilistic and non-deterministic terms avoids us to deal with probabilities in the external choice operator, and thus, when describing the behaviour of probabilistic systems, all the probabilistic choices will be specified by means of probabilistic terms, with their components prefixed by the special action \( i \).

• \( i; N \): This represents the passage of one unit of time, and afterwards this (probabilistic) process behaves like the non-deterministic term \( N \).

We will mainly work with closed and guarded terms, i.e., all the identifiers (\( X \)) are bounded by a \( \mu X \) (there are no free identifiers), and every occurrence of \( X \) within the body of a recursion must be prefixed. We will denote by \( \text{NPROC} \) the set of closed and guarded non-deterministic processes, and by \( \text{PPROC} \) the set of closed and guarded probabilistic processes. We will also say that a term is regular if there is no free identifier in a subterm affected by either a parallel or a hiding operator.

Next, we define the \( \text{age}_1 \) operator, which reflects the passage of one unit of time for non-deterministic processes, when no action is executed. As we will see, this may only occur if no urgent action can be executed at the current instant.

**Definition 1.** The operator \( \text{age}_1 \) is defined by:

\[
\text{age}_1 : \text{NPROC} \rightarrow \text{NPROC}
\]

- \( \text{age}_1(\text{stop}) = \text{stop} \)
- \( \text{age}_1(a(t_1, t_2); P) = a(t_1 - 1, t_2 - 1); P \) if \( t_2 > 0 \) where \( t_1 - 1 = t_1 - 1, \) if \( t_1 > 0, \) and \( 0 - 1 = 0. \)
- \( \text{age}_1(a(0, 0); P) = \text{stop} \)
- \( \text{age}_1(a; P) = a; P \)
- \( \text{age}_1(N_1 \downarrow N_2) = \text{age}_1(N_1) \downarrow \text{age}_1(N_2) \)
- \( \text{age}_1(N_1 \|_A N_2) = \text{age}_1(N_1) \|_A \text{age}_1(N_2) \)
- \( \text{age}_1(\mu X.N) = \mu X.\text{age}_1(N) \)
- \( \text{age}_1(N \setminus a) = \text{age}_1(N) \setminus a. \)

It is immediate to check that this operator is well defined. Let us observe the point 3 of this definition, it establishes that for the timed prefix, once the deadline has expired, if the corresponding action has not been executed, we become deadlocked (the action cannot be executed ever after). However, the point 4 states that urgent actions may wait for the environment to require its execution, although we will see that when some urgent actions are enabled time cannot elapse and a bag of actions containing a maximal number of urgent actions must be executed.

**Example 1.** Let \( N = a(1, 2); i; \text{stop}\|_A a; i; \text{stop} \). The action \( a \) belongs to the synchronisation set, and the first component cannot execute it at the current instant, so one unit of
time must elapse before executing this action. Then, we have:

\[ \text{age}_1(N) = a \langle 0, 1 \rangle ; i; \text{stop} \parallel \{a\} a; i; \text{stop} \]

And now the action \( a \) is immediately executed, because it is urgent, and no time can elapse before doing so.

3.3. Bags

In order to define our operational semantics we need to consider two sets of pairs of bags for every guarded and closed non-deterministic process, \( \text{Bags}(N) \) and \( \text{Now}(N) \). Their elements are pairs \((B, U)\), where \( B \) is a bag of actions and \( U \subseteq B \) is the bag of urgent actions in \( B \). The intuitive interpretation is that \( \text{Bags}(N) \) consists of all bags of actions that \( N \) could execute without any restrictions, and \( \text{Now}(N) \) (feasible bags in \( \text{Bags}(N) \)) only contains those bags having a maximal number of urgent actions. These bags may include a new internal action \( /a \), to deal with hidden actions. This action has only a semantic nature, and thus, it cannot appear in the syntactic description of processes.

**Definition 2.** Let \( N \in \text{NPROC} \) and \( \text{Act}_\tau = \text{Act} \cup \{\tau\} \). We define the set \( \text{Bags}(N) \), as follows:

\[
\text{Bags} : \text{NPROC} \rightarrow \mathcal{P}(\mathcal{B}(\text{Act}_\tau) \times \mathcal{B}(\text{Act}_\tau))
\]

- \( \text{Bags}(\text{stop}) = \{ (\emptyset, \emptyset) \} \).
- For the non-urgent prefix:
  \[
  \text{Bags}(a(t_1, t_2); P) = \begin{cases} 
  \{ (\emptyset, \emptyset) \} & \text{if } t_1 > 0 \\
  \{ (\{1.a\}, \emptyset) \} & \text{if } t_1 = 0 
  \end{cases}
  \]
- \( \text{Bags}(a; P) = \{ (\{1.a\}, \{1.a\}) \} \)
- \( \text{Bags}(N_1 \sqcap N_2) = \text{Bags}(N_1) \cup \text{Bags}(N_2) \)
- \( \text{Bags}(N_1 \parallel_A N_2) = \text{Bags}(N_1) \oplus_A \text{Bags}(N_2) \)
- \( \text{Bags}(N \setminus a) = \bigcup_{(B, U) \in \text{Bags}(N)} (B[\tau/a], U[\tau/a]) \)
  where \( B[\tau/a] \) (resp. \( U[\tau/a] \)) is the bag obtained by replacing every instance of \( a \) in \( B \) with \( \tau \)
- \( \text{Bags}(\mu X.N) = \text{Bags}(N\{\mu X.N/X\}) \)

This definition hardly requires any explanations, except for the parallel and the recursion operators. Bags for the parallel operator are essentially the union of the bags of both components, but taking into account that they must synchronise for the actions in \( A \), and thus for these actions we take the intersection of both bags (see the definition of \( +_A \)). On the other hand, we must take into account the urgency expansion for the parallel operator to obtain each bag of urgent actions, i.e., when one action is urgent on one side, and it can be executed by the other component, this action becomes urgent for the complete process (see the definition of \( <_A \)). With respect to the recursion operator, let us observe that \( \text{Bags}(N\{\mu X.N/X\}) \) is well defined, because we are only considering guarded and closed processes.
Table 1
Non-deterministic transition rules

\[ \begin{array}{ll}
\text{(N1)} & \frac{\phi(N) = 0}{N \xrightarrow{i} \text{age}_1(N)} \\
\text{(N2a)} & \frac{a(0, t)}{P \xrightarrow{\{1.a, 1.a\}} P} \\
\text{(N2b)} & \frac{a; P \xrightarrow{\{1.a, 1.a\}} P}{P} \\
\text{(N3a)} & \frac{N_1 \xrightarrow{(B, U)} P_1}{N_1 \parallel N_2 \xrightarrow{(B, U)} P_1} \\
\text{(N3b)} & \frac{N_2 \xrightarrow{(B, U)} P_2}{N_1 \parallel N_2 \xrightarrow{(B, U)} P_2} \\
\text{(N4a)} & \frac{N_1 \xrightarrow{(B, U)} P_1, B \downarrow A = \emptyset}{N_1 \parallel A N_2 \xrightarrow{(B, U)} P_1 \parallel_A i; \text{age}_1(N_2)} \\
\text{(N4b)} & \frac{N_2 \xrightarrow{(B, U)} P_2, B \downarrow A = \emptyset}{N_1 \parallel A N_2 \xrightarrow{(B, U)} i; \text{age}_1(N_1) \parallel_A P_2} \\
\text{(N4c)} & \frac{N_1 \xrightarrow{(B_1, U_1)} P_1, N_2 \xrightarrow{(B_2, U_2)} P_2, B_1 \downarrow A = B_2 \downarrow A}{N_1 \parallel A N_2 \xrightarrow{(B, U)} P_1 \parallel_A P_2} \\
\text{(N5)} & \frac{N \xrightarrow{\mu X. N / X} (B, U)}{\mu X. N \xrightarrow{(B, U)} P} \\
\text{(N6)} & \frac{N \xrightarrow{(B, U)} P}{N \xrightarrow{\mu \alpha[a]} P \xrightarrow{\mu \alpha[a]} P} \\
\end{array} \]

where \( B = B_1 +_A B_2 \) and \( U = (U_1 \triangleleft A B_2 + U_2 \triangleleft A B_1) \triangleleft_A (B_1 +_A B_2) \)

Definition 3. Let \( N \in \text{NPROC} \). We define:

- The urgency degree of \( N \) by:
  \[ \phi(N) = \text{Max} \{ \text{Card}(U) \mid (B, U) \in \text{Bags}(N) \} \]

- The set \( \text{Now}(N) \) (feasible bags of \( N \)) by:
  \[ \text{Now} : \text{NPROC} \rightarrow \mathcal{P}(\mathcal{B}(\text{Act}_1) \times \mathcal{B}(\text{Act}_1)) \]
  \[ \text{Now}(N) = \{ (B, U) \in \text{Bags}(N) \mid \phi(N) = \text{Card}(U) \} \]

In consequence, \( \text{Now}(N) \) consists of the pairs \( (B, U) \in \text{Bags}(N) \) having the maximum number of urgent actions.

Example 2. The following processes illustrate the application of the functions \( \text{Bags} \) and \( \text{Now} \):

1. \( N_1 = a; i; \text{stop} \parallel b; i; \text{stop} \).
   Both actions \( a \) and \( b \) are urgent, but only one of them can be executed. Thus, we have:
   \begin{align*}
   \text{Bags}(N_1) &= \{(\{1.a\}, \{1.a\}), (\{1.b\}, \{1.b\})\} \\
   \text{Now}(N_1) &= \text{Bags}(N_1) \\
   \phi(N_1) &= 1
   \end{align*}
(2) $N_2 = a; i; \text{stop}\square b(0, 4); i; \text{stop}$.
   In this case only the action $a$ can be executed, because it is urgent:
   \[
   \begin{align*}
   &\text{Bags}(N_2) = \{(1.a), (1.a), (1.b), \emptyset\} \\
   &\text{Now}(N_2) = \{(1.a), (1.a)\} \\
   &\varphi(N_2) = 1
   \end{align*}
   \]

(3) $N_3 = N_1 \|_{\{a\}} N_2$.
   According to the criterion of executing as many urgent actions as possible, we can only execute the bag containing two instances of the action $a$:
   \[
   \begin{align*}
   &\text{Bags}(N_3) = \{(2.a), (2.a), (1.a), (1.a), (1.b), (1.b), (\emptyset, \emptyset)\} \\
   &\text{Now}(N_3) = \{(2.a), (2.a)\} \\
   &\varphi(N_3) = 2
   \end{align*}
   \]

(4) $N_4 = N_3 \|_{\{a\}} c; i; \text{stop}$.
   For this process $\text{Bags}(N_4) = \{(1.c), (1.c), (1.b), (1.b), (\emptyset, \emptyset), (1.b, 1.c), (1.b, 1.c)\}$ and $\varphi(N_4) = 2$. Hence, $\text{Now}(N_4) = \{(1.b, 1.c), (1.b, 1.c)\}$, which is different from $\text{Now}(N_3)$, because the action $a$ belongs to the synchronisation set.

(5) $N_5 = \mu X.a; i; \text{stop}\|_{\{a\}} (a(0, 2); i; X \square b; i; \text{stop})$
   Now we have:
   \[
   \begin{align*}
   &\text{Bags}(N_5) = \{\emptyset, \emptyset, (1.b), (1.b), (1.a), (1.a)\} \\
   &\text{Now}(N_5) = \{(1.b), (1.b), (1.a), (1.a)\} \\
   &\varphi(N_5) = 1
   \end{align*}
   \]

4. Operational semantics

As usual, we present the operational semantics by means of a labelled transition system (LTS), defined by using a set of rules. Then, the operational semantics will be a restriction of the LTS thus obtained, imposing the execution of a maximal number of urgent actions at each step.

The LTS consists of two types of transitions: non-deterministic and probabilistic transitions. Non-deterministic transitions represent the evolution for a non-deterministic term either by executing a bag or by ticking one unit of time without executing any actions, so they can adopt two forms:

- $N \rightarrow N'$ represents the passage of time (one tick).
- $N \xrightarrow{(B, U)} P$ represents the evolution of $N$ by executing the bag $B$, where $(B, U) \in \text{Bags}(N)$, and $B \neq \emptyset$.

On the other hand, probabilistic transitions represent the internal probabilistic decisions that the system makes. No actions are then executed on these transitions, but the resolution of a conflict takes always one unit of time. They adopt the following form:

- $P \xrightarrow{i} q N$ where $q \in [0, 1]$

which means that the probabilistic process $P$ behaves as the non-deterministic process $N$ with probability $q$ (one unit of time later).
4.1. Non-deterministic transition rules

These rules are presented in Table 1. First rule establishes that time can elapse without executing any actions only when there are no urgent actions. Let us observe that this rule can be applied to \texttt{stop}, and in fact, it is the only rule applicable to it. Rules N2a and N2b capture the semantics of the timed prefix and the urgent action prefix, in both cases the bag \{1.a\} can be executed. N3a and N3b define the semantics of the external choice, according to the interpretation given in Section 3.2. N4a, N4b and N4c define the behaviour for the parallel composition. Concretely, with N4a and N4b we establish the autonomous evolution of each component, by executing a bag not including actions in the synchronisation set, and the passage of one unit of time for the other component, which must be now a probabilistic term (thus we need to include the action \(i\) on this term). Rule N4c defines the parallel evolution of both components, each one executing a bag of actions, but synchronising on the actions in the synchronisation set \(A\); thus, we impose both bags to have the same number of instances of actions in \(A\). The behaviour of recursive processes is captured by rule N5, and finally rule N6 defines the behaviour of the hiding operator, transforming the hidden action into \(\tau\) (but preserving its urgent character, if it was the case).

4.2. Probabilistic transition rules

Table 2 summarises the probabilistic transition rules. Rule P1 represents the passage of one unit of time, making a trivial decision. Rules P2a and P2b establish the expected behaviour of the probabilistic choice, while rule P3 establishes the evolution for a parallel composition of probabilistic processes, by executing a unique action \(i\), once the probabilistic decisions on both components have been resolved. Finally, rule P4 shows the way we deal with the hiding operator for probabilistic terms, expanding it down to the non-deterministic process obtained once the probabilistic choice has been resolved.

**Definition 4.** Let \(\mathcal{T}(R)\) be the lts of a closed and guarded process \(R \in TPAL_p\), obtained by applying the rules. We define the Operational Semantics, \(\mathcal{S}(R)\), by restricting \(\mathcal{T}(R)\) in

<table>
<thead>
<tr>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic transition rules</td>
</tr>
<tr>
<td>(P1) (i; N \xrightarrow{}_{1.0} N)</td>
</tr>
<tr>
<td>(P2a) (\frac{P_1 \xrightarrow{i} r N_1 P_2 \xrightarrow{1-q} _q N_1}{P_1 + q P_2 \xrightarrow{i} _r N_1})</td>
</tr>
<tr>
<td>(P2b) (\frac{P_1 \xrightarrow{i} r N_2 P_2 \xrightarrow{(1-q)r} _{1-q} N_2}{P_1 + q P_2 \xrightarrow{i} _r N_2})</td>
</tr>
<tr>
<td>(P3) (\frac{P_1 \xrightarrow{i} _{q_1} N_1 P_2 \xrightarrow{i} _{q_2} N_2}{P_1 \parallel_A P_2 \xrightarrow{i} _{q_1,q_2} N_1 \parallel_A N_2})</td>
</tr>
<tr>
<td>(P4) (\frac{P \xrightarrow{i} _q N \backslash a \xrightarrow{i} _q N \backslash a}{P \cap a \xrightarrow{i} _q N \backslash a})</td>
</tr>
</tbody>
</table>
the following way:

\[
S(R) = \{ P \xrightarrow{i,q} N \mid P \xrightarrow{i,q} N \in T(R) \} \cup \{ N \xrightarrow{i} N' \mid N \xrightarrow{i} N' \in T(R) \}
\]

\[
\cup \{ N \xrightarrow{(B,U)} P \mid P \xrightarrow{(B,U)} N \in T(R), (B,U) \in \text{Now}(N) \}
\]

The definition of the timed-probabilistic computations is now straightforward. They have the following form:

\[
R \xrightarrow{s} q R'
\]

where \( R, R' \in \text{TPAL}_p, q \in [0, 1] \), and \( s \) is a trace, which is an alternating sequence of bags and ticks (any number of i-actions).

We can easily obtain the following two properties for the complete lts, \( T(N) \). Their proofs can be found in Appendix A.

**Proposition 1.** Let \( B \) be a non-empty bag, and \( U \subseteq B \). If \( N \xrightarrow{(B,U)} P \in T(N) \), then for every \( B' \subseteq B, B' \neq \emptyset \), there exists a probabilistic process \( P' \) such that \( N \xrightarrow{(B',B'\cup U)} P' \in T(N) \).

**Proposition 2.** Let \( N \in \text{NPROC}, (B, U) \in \text{Bags}(N) \), with \( B \neq \emptyset \). Then, there exists a probabilistic process \( P \) such that \( N \xrightarrow{(B,U)} P \in T(N) \).

### 5. Sequential components

We use a similar approach to that followed in [26] in order to split every closed and guarded \( \text{TPAL}_p \) process into a set of sequential and decision components.

**Definition 5 (Sequ).** We define the set of sequential components, \( \text{Sequ} \), by the following BNF notation:

\[
S :::= \text{stop} \mid a(t_1, t_2); P \mid S \| A \mid A \| S \mid S \Box a \mid a; P \mid S \setminus a
\]

where \( P \in \text{PPROC} \).

**Definition 6.** We define the following function: \( \text{dex} : \text{NPROC} \rightarrow \mathcal{P}(<\text{Sequ}>) \)

For any \( N \in \text{NPROC} \), \( \text{dex}(N) \) is the set of sequential components of \( N \), and it is defined as follows:

- \( \text{dex}(\text{stop}) = \text{stop} \)
- \( \text{dex}(a(t_1, t_2); P) = a(t_1, t_2); P \)
- \( \text{dex}(a; P) = a; P \)
- \( \text{dex}(N_1 \| A N_2) = \text{dex}(N_1) \| A \cup A \| \text{dex}(N_2) \)
- \( \text{dex}(N_1 \Box N_2) = \text{dex}(N_1) \Box \text{dex}(N_2) \)
• \( \text{dex}(N \setminus a) = \text{dex}(N) \setminus a \)
• \( \text{dex}(\mu X.N_1) = \text{dex}(N_1[\mu X.N_1/X]) \)

In this definition, \( \| A \cdot A \cdot A \cdot A \cdot A \) and \( \setminus a \) have been generalised to operate on sets of sequential components:

\[
\begin{align*}
P_1 \square P_2 &= \{ L_1 \square L_2 \mid L_1 \in P_1, L_2 \in P_2 \} \\
P \setminus a &= \{ L \setminus a \mid L \in P \} \\
P \| A &= \{ L \| A \mid L \in P \} \\
A \| P &= \{ A \| L \mid L \in P \}
\end{align*}
\]

From the previous definition we can see that \( \text{dex} \) essentially acts by splitting a parallel term into the arguments of the parallel operator. Furthermore, in the case of a choice, it takes the sequential components of its arguments, it generates the Cartesian product, and then, the sequential components of the original term are the result of combining these pairs with the external choice operator. Both cases are illustrated by the following example.

**Example 3.**
(1) \( \text{dex}((a(0,7); i; b(0, \infty); \text{stop}) \|_{\{b\}}(\mu X.c(3, 3); i; b; i; X)) \)
\[
= \{ (a(0,7); i; b(0, \infty); \text{stop}) \|_{\{b\}}, \\
    \{b\}(c(3, 3); i; b; i; \mu X.c(3, 3); i; b; i; X) \}
\]
(2) \( \text{dex}((a; i; \text{stop}) \|_{\{a\}} b; i; a; i; \text{stop}) \square (c; i; \text{stop}) \|_{\{c\}} c(1, 1); i; \text{stop}) \)
\[
= \{ (a; i; \text{stop}) \|_{\{a\}} \square (c; i; \text{stop}) \|_{\{c\}}, \\
    (a; i; \text{stop}) \|_{\{a\}} \square (c; i; \text{stop}) \|_{\{c\}}, \\
    (b; i; a; i; \text{stop}) \square (c; i; \text{stop}) \|_{\{c\}}, \\
    (b; i; a; i; \text{stop}) \square (c; i; \text{stop}) \|_{\{c\}} \}
\]

**Definition 7.** A set of sequential components \( P \) is said to be complete if there exists \( Q \in \text{NPROC} \), such that \( \text{dex}(Q) = P \).

**Definition 8.** We define the complexity degree for any \( \text{NPROC} \) term by
\[
\gamma(\text{stop}) = \gamma(a(t_1, t_2); P) = \gamma(a; P) = 0 \\
\gamma(N \square N_2) = \gamma(N_1 \|_A N_2) = 1 + \text{Max}\{\gamma(N_1), \gamma(N_2)\} \\
\gamma(N \setminus a) = \gamma(\mu X.N) = 1 + \gamma(N)
\]

**Proposition.**
(1) \( \text{dex} \) is well defined
(2) \( \text{dex} \) is not injective.

**Proof.**
(a) Immediate, by induction on the complexity degree of \( N \in \text{NPROC} \).
(b) Consider the following counterexample:
\[
\text{dex}(a; i; \mu X.a; i; X) = \text{dex}(\mu X.a; i; X)
\]

\( \text{Seq} \) does not contain probabilistic components, we therefore need to define a probabilistic extension, \( \text{SeqDec} \), which is the set of sequential and decision components. Afterwards, we define the function \( \text{adv} \), which extends \( \text{dex} \) for probabilistic terms.
**Definition 9.** SeqDec is defined by:

\[
\text{SeqDec} ::= \text{Sequ} \mid \text{Dec}
\]

\[
\text{Dec} ::= D \mid D \\ A \mid A \\ D \mid D \setminus a
\]

with \(D = \{\tau; P_1 +_p \tau; P_2 \mid P_1, P_2 \in \mathcal{P}(\text{SeqDec}), P_1 \neq \emptyset, P_2 \neq \emptyset, p \in [0, 1]\}\).

**Definition 10.** We define the function \(\text{adv}\) as follows:

\[
\text{adv} : \mathcal{PProc} \rightarrow \mathcal{P}(\text{SequDec})
\]

- \(\text{adv}(i; N) = \text{dex}(N)\)
- \(\text{adv}(P \parallel_A P) = \text{adv}(P) \cup_A \text{adv}(P)\)
- \(\text{adv}(P \setminus a) = \text{adv}(P) \setminus a\)
- \(\text{adv}(P_1 +_q P_2) = \{\tau; \text{adv}(P_1) +_q \tau; \text{adv}(P_2)\}\)

It is immediate to check that \(\text{adv}\) is well defined, by structural induction on \(\mathcal{PProc}\) terms.

This function can also be extended in a straightforward way to any \(\mathcal{P} = \{P_1, \ldots, P_n\} \in \mathcal{P}(\mathcal{PProc}), \mathcal{P} \neq \emptyset\), as follows:

\[
\text{adv}(\mathcal{P}) = \bigcup_{i=1}^{n} \text{adv}(P_i)
\]

**Example 4.** The following terms illustrate how \(\text{adv}\) works:

1. \(P_1 = i; a; i; \text{stop} +_{0.2} i; b; i; \text{stop}\)
   \[
   \text{adv}(P_1) = \{\tau; \{a; i; \text{stop}\} +_{0.2} \tau; \{b; i; \text{stop}\}\}
   \]
2. \(P_2 = (i; (a; i; \text{stop})\parallel_{[a]} b; i; a; \text{stop})) +_{0.3} (i; \text{stop})\)
   \[
   \text{adv}(P_2) = \{\tau; \{a; i; \text{stop}\parallel_{[a]} b; i; a; \text{stop}\} +_{0.3} \tau; \{\text{stop}\}\}
   \]
3. \(P_3 = P_1 \parallel_{[b]} P_2\)
   \[
   \text{adv}(P_3) = \{\tau; \{a; i; \text{stop}\} +_{0.2} \tau; \{b; i; \text{stop}\}\parallel_{[b]},\ 
   \ 
   \parallel_{[b]} \{\tau; \{a; i; \text{stop}\parallel_{[a]} b; i; a; \text{stop}\} +_{0.3} \tau; \{\text{stop}\}\}\}
   \]

6. **Net model**

In this section we define the particular class of timed-probabilistic Petri nets that we will use to translate regular, closed and guarded TPAL\(_p\) terms. Specifically, we will use Timed-Probabilistic Arc Petri Nets (TPNets), which have their tokens annotated with an age (a non-negative integer value indicating the elapsed time from its creation) and some arcs connecting places with transitions have associated a time interval, which limits the age of the tokens to be consumed to fire the adjacent transition. In the particular model that we consider in this paper some transitions can be urgent, in the sense that no time can elapse once they are enabled. Probabilities are included by means of the so called decision places, for which all their outgoing arcs must be labelled with a probability.
The interpretation and use of TP Nets can be obtained from a collection of processes interacting with one another according to a rendez-vous mechanism. Each process may execute either local or synchronisation actions. Local actions are those that the process may execute without cooperation from another process, and thus in the Petri net model of the whole system they would appear as transitions with a single precondition place, while synchronisation actions would have several precondition places, which correspond to the states at which each one of the involved processes being ready to execute the action. Then, each time interval establishes some time restrictions related to a particular process (for instance the time that a local processing may require). In consequence, the firing of a synchronisation action can be done in a time window, which depends on the age of the tokens on its precondition places.

6.1. Definitions

The following definitions capture in a formal way all the aspects of TP Nets that we introduced in Section 2.2.

**Definition 11** (Net model). A Timed-Probabilistic Arc Petri net (TPNet) is a tuple \( N = (P, T, F, \lambda, I, r) \), where:

- \( P \) is the set of places, which consists of two components: \( P = P_1 \cup P_2 \), with \( P_1 \cap P_2 = \emptyset \).
  - Places of \( P_2 \) will correspond to probabilistic decisions.
- \( T \) is the set of transitions, which consists again of two components: \( T = T_1 \cup T_1^* \), such that: \( P \cap T = \emptyset \). \( T_1 \cap T_1^* = \emptyset \), \( \lambda(t) \neq \emptyset \) and \( \lambda(t) \neq \emptyset \) \( \forall t \in T \). Transitions in \( T_1 \) are called non-urgent, and transitions in \( T_1^* \) are called urgent.
- \( F \) is the flow relation, \( F \subseteq (P \times T) \cup (T \times P) \), such that:
  - \( \forall p \in P_2, \forall t \in p^* : t \in T_1^* \land \lambda(t) = \{ p \} \).
  - \( \lambda \) is the transition labelling function: \( \lambda : T \rightarrow \text{Act}_\tau \), such that \( \forall p \in P_2, \forall t \in p^*, \lambda(t) = \tau \).
  - \( I \) is the arc labelling function: \( I : F|_{P_1 \times T} \rightarrow (\mathbb{N} \times \mathbb{N} \cup \{ \infty \}) \cup \{ \ast \} \), such that \( I(p, t) \in \mathbb{N} \times \mathbb{N} \cup \{ \infty \}, \forall (p, t) \in P_1 \times T_1 \).
- \( r \) gives the probabilities for the arcs in \( P_2 \times T_1^* \).
  - \( r : F|_{P_2 \times T_1^*} \rightarrow (0, 1] \), such that \( \forall p \in P_2, \sum_{t \in p^*} r(p, t) = 1 \)

We only consider T-restrictive nets \((\ast \neq \emptyset \land \ast' \neq \emptyset, \forall t \in T)\), because we have an urgent step semantics, i.e., a semantics that only permits the execution of those enabled multisets of transitions containing a maximal number of urgent transitions. Further restrictions are introduced to fix more precisely the particular class of nets that we will obtain from the translation. More specifically, postcondition transitions of decision places will be always internal and urgent, and these latter transitions cannot have any other precondition places. However, notice that those urgent transitions without predecessors in \( P_2 \) can have several precondition places (they can be obtained as result of a synchronisation).

**Definition 12** (Markings). Let \( N = (P, T, F, \lambda, I, r) \) be a TP Net. A marking \( M \) of \( N \) is a function \( M : P \rightarrow B(\mathbb{N}) \). \( M(p)(n) \) is the number of tokens with age \( n \) in \( p \), supposing that \( M(p)(n) = 0 \) for nearly all \( n \) (on each place there will be a finite number of tokens). A marking \( M_0 \) can be initial if and only if \( M_0(p)(n) = 0, \forall n > 0, \forall p \in P \), and we will also
say that a marking is \textit{stable} if there is no marked place in \( P_2 \). \( M(N) \) will denote the set of markings of \( N \). Besides, \( \forall p \in P_2, \forall n > 0 \), we will have \( M(p)(n) = 0 \), i.e., the tokens on probabilistic decision places cannot grow up.

Given a marking \( M \) of \( N \), and \((p, t) \in P_1 \times T\), we define: \( M |_{I(p,t)} : P \rightarrow B(N) \), by

\[
M |_{I(p,t)}(p)(n) = \begin{cases} 
M(p)(n) & \text{if } I(p,t) \in \mathbb{N} \times (\mathbb{N} \cup \{\infty\}), n \in I(p,t) \\
M(p)(n) & \text{if } I(p,t) = * \\
0 & \text{otherwise}
\end{cases}
\]

Given a \( TPNet \) \( N = (P, T, F, I,r) \), and a marking \( M \) of \( N \), we will say that \((P, T, F, I, r, M)\) is a \textit{Marked TPNet}, and we will denote by \( MTPNet \) the set of marked \( TPNets \).

6.2. Firing rule

The behaviour of \( TPNets \) is governed by the following policy: when one or several decision places are marked, the net must evolve by resolving the corresponding probabilistic decisions, before executing any other transitions. Thus, for non-stable markings the only possible evolutions are those obtained by executing multisets of transitions in \( T_1^* \cap P_2^* \), and multisets of transitions in \( P_1^* \) are only enabled at stable markings. Furthermore, we always impose for the firing of a multiset of transitions that it must contain a maximal number of urgent transitions (we will say that it is \textit{u-maximal}).

6.2.1. Activation

\textbf{Definition 13.} Let \( N = (P, T, F, \lambda, I, r) \) be a \( TPNet \), and \( M \) a non-stable marking of it. We say that a multiset of transitions \( R \) in \( T_1^* \cap P_2^* \) is \textit{enabled} at \( M \), which will be denoted by \( M[R] \), if and only if:

\[ \forall p \in P_2 : \text{Card}(M(p)) \geq \sum_{t \in p^*} R(t) \]

Thus, for every instance of each transition \( t \) in \( R \) we need one different token on its (unique) precondition place.

\textbf{Definition 14.} Let \( N = (P, T, F, \lambda, I, r) \) be a \( TPNet \), and \( M \) a stable marking of it. We say that a multiset of transitions \( R \) in \( T \cap P_1^* \) is \textit{enabled} at \( M \), which will be denoted by \( M[R] \), if and only if for every \( p \in P_1 \) there is a function \( \Phi_p : Enum(B_p) \rightarrow Enum(M(p)) \), where \( B_p \) is the multiset over \( (\mathbb{N} \times (\mathbb{N} \cup \{\infty\})) \cup \{*\} \) defined by:

\[ B_p = \{R(t).I(p,t) \mid t \in p^*\} \]

such that for every \( \Phi_p(k, I(p,t)) = (i, age_j) \) we have either \( I(p,t) = '\infty' \) or \( age_j \in I(p,t) \), when \( I(p,t) \) is an interval.

Thus, for every instance of each transition \( t \) in \( R \) we need one token on each precondition place \( p \) of \( t \), such that when \( I(p,t) \neq '\infty' \) its age must be in the time interval \( I(p,t) \).
Definition 15. Let $N = (P, T, F, \lambda, I, r)$ be a TPNet, $M$ a marking of it, and $R$ an enabled multiset of transitions at $M$. We say that $R$ is $u$-maximal if there is no multiset of transitions $R'$, such that $M[R'] \land \sum_{t \in T_1^*} R'(t) \geq \sum_{t \in T_1^*} R(t)$.

6.2.2. Time elapsing

Time elapsing is only possible for stable markings, and it is captured in two ways: firstly when we fire a u-maximal bag we consider that one tick of time has elapsed, but we also allow time elapsing if no u-maximal bag is enabled at the current marking. In the latter case, time can elapse until a u-maximal bag becomes enabled.

Definition 16 (Aging). Given $N = (P, T, F, \lambda, I, r) \in TPNet$, and $M$ a stable marking of it, we say that one tick can elapse in $N$ at $M$ if there is no enabled multiset of transitions containing urgent transitions.

The new marking that we obtain is $M' = age_1(M)$, where $age_1$ increases the age of each token in $M$ in one unit:

$$\forall p \in P, \quad age_1(M)(p)(n) = \begin{cases} 0 & \text{if } n = 0 \\ M(p)(n-1) & \text{if } n > 0 \end{cases}$$

These evolutions are represented by $M[\emptyset]_{1.0}M'$.

6.2.3. Firing bags of transitions

Definition 17. Given a TPNet $N = (P, T, F, \lambda, I, r)$, $M$ a non-stable marking of it, and $R$ an enabled u-maximal multiset of transitions in $T_1^* \cap P_2^*$. With the firing of $R$ a new marking $M'$ is generated, defined by

$$M'(p) = (M(p) - B_-(p)) + B_+(p)$$

where

$$B_+(p) = \bigcup_{t \in \cdot p} R(t).0, \quad B_-(p) = \bigcup_{t \in \cdot p} R(t).0.$$ 

We will denote this step by $M[R]_qM'$, with $q = \prod_{t \in \cdot p} r(p,t)^{R(t)}$.

With these probabilistic evolutions no time elapses. Notice that every token that we use to fire $R$ must have age 0, since we do not allow the passage of time when these transitions are enabled. Furthermore, the new tokens that appear on every postcondition place of the transitions in $R$ have age 0.

Definition 18. Given a TPNet $N = (P, T, F, \lambda, I, r)$, $M$ a stable marking of it, and $R$ an enabled u-maximal multiset of transitions in $P_1^*$. With the firing of $R$ a new marking $M'$ is generated, defined by

$$M'(p) = age_1(M(p) - B_-(p)) + B_+(p)$$

where $B_+(p) = \bigcup_{t \in \cdot p} R(t).0$, which are the new tokens generated by this firing, and $B_-(p)$ are the tokens in $p$ selected for the firing of $R$ according to Definition 14.
This step will be denoted by $M[R]_1 M'$.

Notice that empty bags are also considered in this definition, in order to capture time elapsing on the net.

**Definition 19 (Step semantics).** Given a $TPNet$ $N = (P, T, F, I, r)$ and $M_0$ an initial marking of it, we say that a sequence $\sigma = M_0[R_0]_{q_0} M_1[R_1]_{q_1} \ldots [R_{n-1}]_{q_{n-1}} M_n$ is a finite probabilistic-timed step sequence (pt-sequence) if and only if $M_i$ is a marking of $N$, $R_i$ is a u-maximal multiset of transitions, $q_i \in [0, 1]$ and $M_i[R_i]_{q_i} M_{i+1}$, for $i = 0, \ldots, n - 1$. These steps will be denoted $M_0[\sigma] M_n$.

Then, we will say that $M \in \mathcal{M}(N)$ is reachable if and only if $\exists \sigma$ pt-sequence such that $M_0[\sigma] M$. As usual, the set of reachable markings will be denoted by $[M_0]$.

The step semantics of $(N, M_0)$ is then defined by

$$S(N, M_0) = \{\sigma \mid M_0[\sigma] M, \text{ with } M \in \mathcal{M}(N)\}$$

7. **Net semantics**

In this section we define the $TPNet$ associated with any regular, closed and guarded process of $TPAL_p$. We first need to define an operational semantics for sequential and decision components.

**Definition 20.** The operational semantics for sequential and decision components is defined by a labelled transition system with two kinds of transitions:

- Non-deterministic transitions, which have the following form:

  $L (a, \phi) \rightarrow P$

  where $L \in \mathcal{P}(Sequ)$, $P \in \mathcal{P}(SequDec)$, $L \neq \emptyset$, $P \neq \emptyset$, $a \in \text{Act}_\tau$, and $\phi : L \rightarrow \Sigma$, with $\Sigma = (\mathbb{N} \times (\mathbb{N} \cup \{\infty\})) \cup \{\#\}$.

  In a transition $L (a, \phi) \rightarrow P$ the set of sequential components $L$ evolves to $P$ by executing the action $a$. The function $\phi$ indicates the time restrictions associated with each component in $L$ in order to perform this transition.

  Non-deterministic transitions are those defined by the rules in Table 3.

- Probabilistic transitions:

  $L (\tau, \#) \rightarrow_P L'$

  where $L \in \mathcal{SequDec}$, $L' \in \mathcal{P}(SequDec)$, and $p \in [0, 1]$. It means that $L$ executes a $\tau$ and evolves to $L'$ with probability $p$.

  Rules defining these probabilistic transitions are presented in Table 4.

  With this operational semantics we may now define the $TPNet$ associated with any regular, closed and guarded $TPAL_p$ process, by considering a place for each possible sequential and decision component, and a transition for each possible evolution according to the operational semantics presented above.
Table 3
Non-deterministic rules for sequential components

(1) Timed prefix: \( \{ a(t_1, t_2); P \} \overset{(a, \phi)}{\rightarrow} adv(P) \)
where \( \phi(a(t_1, t_2); P) = (t_1, t_2) \)

(2) Urgent prefix: \( \{ a; P \} \overset{(a, \phi)}{\rightarrow} adv(P) \)
where \( \phi(a; P) = * \)

(3) Parallel composition (asynchronous):
\[
L \overset{(a, \phi)}{\rightarrow} P \quad A \| (A \cup A \| L \overset{(a, \phi)}{\rightarrow} P) \]
with \( \phi(L \| A) = \phi(L) \), \( \phi'(L \| A) = \phi(L) \), \( \forall L \in \mathbb{L} \)

(4) Parallel composition (synchronising):
\[
L_1 \overset{(a, \phi)}{\rightarrow} P_1, \quad L_2 \overset{(a, \phi)}{\rightarrow} P_2 \quad (L_1 \| A | A | L_2 \overset{(a, \phi)}{\rightarrow} P_1 \| A \| A \| P_2) \]
with \( \phi(L_1 \| A) = \phi(L_1) \), \( \forall L_1 \in L_1 \); \( \phi(A \| L_2) = \phi_2(L_2) \), \( \forall L_2 \in L_2 \)

(5) Non-deterministic choice:
\[
L_1 \cup L_2 \overset{(a, \phi)}{\rightarrow} P \quad L_1 \| (L_2 \| A \| L_2 \overset{(a, \phi)}{\rightarrow} P) \]
where \( \phi'(L) = \begin{cases} \phi(L) & \text{if } L \in L_1 \\ \phi(L_2) & \text{if } L = L_2 \| R \in L_2 \| R \end{cases} \)
\( \phi''(L) = \begin{cases} \phi(L) & \text{if } L \in L_1 \\ \phi(L_2) & \text{if } L = R \| L_2 \in R \| L_2 \end{cases} \)
\( L_1, L_2, R, P \in \text{Sequ}, \ L_1 \cap L_2 = \emptyset, \ R \text{ complete}, \ L_1 \text{ could be empty.} \)

(6) Hiding:
\[
L \overset{a, \phi}{\rightarrow} P \quad L \| a \overset{a, \phi}{\rightarrow} P \| L \quad L \| a \overset{b, \phi}{\rightarrow} P \| L \quad L \| a \| b, \phi \overset{b}{\rightarrow} P \| L \| a, \text{ with } a \neq b \]
where \( \phi'(L \| a) = \phi(L) \), \( \forall L \in \mathbb{L} \)

Table 4
Probabilistic rules for decision components

(1) Probabilistic choice:
\[
\tau; \overset{P_1 + \tau; P_2 \overset{(a, s)}{\rightarrow} p}{P_1} \quad \tau; \overset{P_1 + \tau; P_2 \overset{(a, s)}{\rightarrow} 1 - p}{P_2} \]

(2) Parallel composition:
\[
L \overset{(a, s)}{\rightarrow} P \quad L \| A \overset{(a, s)}{\rightarrow} P \| A \quad A \| L \overset{(a, s)}{\rightarrow} P \| A \| P \]

(3) Hiding:
\[
L \overset{(a, s)}{\rightarrow} P \quad L \| a \overset{(a, s)}{\rightarrow} P \| L \]
Definition 21 (TPNet semantics). Let $RTPAL_p$ be the set of regular, closed and guarded terms of $TPAL_p$. We define a mapping:

$$\text{Net} : RTPAL_p \longrightarrow \text{MTPNet}$$

Given $R \in RTPAL_p$, $\text{Net}[R] = (P, T, F, \lambda, I, r, M_0)$, where

- $P = \text{Sequ} \cup \text{Dec}$, and according to Definition 11 we take $P_1 = \text{Sequ}$ and $P_2 = \text{Dec}$.
- $T = T_{Ndet} \cup T_{Dec}$, where:
  $$T_{Ndet} = \{(\emptyset, u, P) \mid L \xrightarrow{(u,\phi)} P\}$$
  $$T_{Dec} = \{(\{L\}, \tau, P) \mid L \xrightarrow{(\tau,\phi)} P\}$$

In order to identify those transitions that are urgent, given a transition $t = (\emptyset, u, P) \in T_{Ndet}$, if there exists $L_i \in L$ such that $\phi(L_i) = \ast$, then this transition is considered to be urgent ($t \in T^*_1$), otherwise $t \in T_1$ (non-urgent). Furthermore, all transitions in $T_{Dec}$ are urgent, i.e., $T_{Dec} \subseteq T^*_1$.

- $F = \{(L, t) \mid t = (\emptyset, u, P) \in T, \text{ and } L \in L\} \cup \{(t, P) \mid t = (\emptyset, u, P) \in T, \text{ with } P \in \mathcal{P}\}$
- $\lambda(\emptyset, u, P) = u$, for all $(\emptyset, u, P) \in T$.
- For all $(L, t) \in F\mid P_1 \times T$ we take $I(L, t) = \phi(L)$, where $t = (\emptyset, u, P)$, $L \in L$ and $L \xrightarrow{(u,\phi)} P$.
- For every $(\{L\}, t) \in F\mid P_2 \times T^*_1$, with $t = (\{L\}, \tau, P)$ and $L \xrightarrow{(\tau,\phi)} P$, we take:

$$r(\{L\}, t) = \begin{cases} p & \text{if } L = \tau; P_1 + p \tau; P_2 \text{ and } P = P_1 \\ 1 - p & \text{if } L = \tau; P_1 + p \tau; P_2 \text{ and } P = P_2 \end{cases}$$

- $M_0$ is defined as follows:

  - $M_0(p)(n) = 0 \forall p \in P \forall n > 0$
  - $M_0(p)(0) = 1$ if $R \in \text{NPROC}, p \in \text{dex}(R)$
  - $M_0(p)(0) = 1$ if $R \in \text{PPROC}, p \in \text{adv}(R)$
  - $M_0(p)(0) = 0$ otherwise

It can be easily checked that the net obtained fulfils the conditions of Definition 11.

Therefore, each sequential and decision component of $R$ is an initial place of $\text{Net}[R]$, and the net structure is obtained by using the transition system obtained from the rules in Tables 3 and 4.

Example 5. In this example we see how we can use the $TPAL_p$ language to model the AUY-protocol, and the corresponding translation to TP Nets. This protocol ensures a reliable communication in a system where channels may fail. The system consists of both a sender and a receiver, communicating messages each other through two unreliable channels, which can lose messages with probability $p \in (0, 1)$. We assume that the transmission delay for both channels is $e$ units of time, and we suppose that when the first channel fails, it replaces the message by a special message $\lambda$ (after a time-out of $t$ units of time), which is sent to the receiver. Once the receiver gets a message, if it is not a $\lambda$, it sends an acknowledgement
message to the sender through the second channel, which can lose it again with probability $p$. In such a case, this channel replaces the $ack$ message by a $\lambda'$ message, which is sent to the sender after $t$ units of time. On the other hand, if the receiver gets a $\lambda$ message, it sends a bad message ($\lambda''$) through the second channel, which causes this channel to generate a $\lambda'$ message for the sender (after $t$ units of time). Then, once the sender gets a $\lambda'$ message, it resends the original message through the first channel.

The specification of this protocol follows:

**Sender:**

\[ T = \mu X. (\text{in}(0, \infty); i; \mu Y. (\text{msg}(e, e); i; (\text{ack}; i; X \square \lambda'; i; Y))) \]

**Channels:**

\[ C_1 = \mu X. (\text{msg}; (i; \lambda(t, t); i; X + p; i; \text{msg'}; i; X)) \]

\[ C_2 = \mu X. ((\text{ack'}; (i; \lambda'(t, t); i; X + p; i; \text{ack}; i; X)) \square (\lambda''; i; \lambda'(t, t); i; X)) \]

**Receiver:**

\[ R = \mu X. ((\text{msg'}(0, \infty); i; \text{out}; i; \text{ack'}(e, e); i; X) \square (\lambda; i; \lambda''; i; X)) \]

**AUY-protocol:**

\[ \text{AUY} = (T \parallel \{\text{msg, ack, } \lambda\}) (C_1 \parallel C_2) \parallel \{\text{msg', ack', } \lambda', \lambda''\} \parallel R \]

We now apply the function $\text{dex}$ to $\text{AUY}$, obtaining four sequential components:

\[ \text{dex}(\text{AUY}) = \{ \text{in}(0, \infty); i; \mu Y. (\text{msg}(e, e); i; (\text{ack}; i; T \square \lambda'; i; Y))) \parallel \{\text{msg', ack', } \lambda', \lambda''\}, \]

\[ \{\text{msg, ack, } \lambda\} \parallel (\text{msg}; (i; \lambda(t, t); i; C_1 + p; i; \text{msg'}; i; C_1)) \parallel \{\text{msg', ack', } \lambda', \lambda''\}, \]

\[ \{\text{msg, ack, } \lambda\} \parallel ((\text{ack'}; (i; \lambda'(t, t); i; C_2 + p; i; \text{ack}; i; C_2)) \square (\lambda''; i; \lambda'(t, t); i; C_2)) \parallel \{\text{msg', ack', } \lambda', \lambda''\}, \]

\[ \{\text{msg', ack', } \lambda', \lambda''\} \parallel (\text{msg'}(0, \infty); i; \text{out}; i; \text{ack'}(e, e); i; R) \square (\lambda; i; \lambda''; i; R)) \} \]

These sequential components are the initial places of the $\text{TPNet}$. Now, from the rules in Tables 3 and 4 the net structure is constructed (Fig. 8).

### 7.1. Safeness

The previous definition always generates a safe $\text{TPNet}$, i.e., we will never get a marking in which a place has two or more tokens. This is because the markings that we can obtain with the firing of transitions in the generated $\text{TPNets}$ are well-formed [26].

**Definition 22.** We say that a set of sequential and decision components $L \in \mathcal{P}(\text{SeqDec})$ is well-formed (wf) if it can be obtained by applying:

1. $\{\text{stop}\}, \{a \langle t_1, t_2\}; P\}$ and $\{a; P\}$ are wf.
2. If $P_1$ and $P_2$ are wf, then $\{\tau; P_1 + q; \tau; P_2\}$ is wf.
3. If $P_1$ and $P_2$ are wf, then $P_1 \parallel A \cup A \parallel P_2$ is wf.
4. If $P_1 \cup P_2$ is wf, with $P_1 \cap P_2 = \emptyset$, then: if $P_1 = \emptyset$ or there is at least one component of $P_1$ that has no choice operator (neither non-deterministic nor probabilistic) at the highest level, then $P_1 \cup (P_2 \square \text{dex}(Q))$ and $P_1 \cup (\text{dex}(Q) \square P_2)$ are wf.
5. If $P$ is wf, then $P \setminus a$ is wf.

We need the following properties to conclude that the $\text{TPNets}$ obtained from Definition 21 are safe.
Proposition 4. Every complete set of sequential components is wf.

Proof. Let $\mathbb{P} = \text{dex}(Q)$ be a complete set of sequential components, with $Q \in \text{RTPAL}_p$. The proof is made by induction on the complexity degree of $Q$:

- **Induction basis:** For $\text{stop}$, $a(t_1, t_2); P$, and $a; P$: Trivial.

- **Induction step:** The reasoning is similar for all the cases, so we only mention two of them:
  
  - $Q = N_1 \square N_2$, then $\text{dex}(Q) = \text{dex}(N_1) \square \text{dex}(N_2)$, with $\gamma(N_i) < \gamma(Q)$, $i = 1, 2$, and by the induction hypothesis $\text{dex}(N_i)$ is wf, $i = 1, 2$. By the definition of wf, it follows that $\text{dex}(Q)$ is wf.
  
  - $Q = \mu X. N$, then $\text{dex}(Q) = \text{dex}(N[\mu X. N/X])$, with $\gamma(N) < \gamma(Q)$. By the induction hypothesis $\text{dex}(N)$ is wf, hence $\text{dex}(Q)$ is wf too. □
Proposition 5. For any $P \in \text{PPROC}$, $\text{adv}(P)$ is wf.

Proof. Immediate, by induction on the structure of $\text{PPROC}$ terms. \qed

Proposition 6. Let $S, Q \in \text{RTPAL}_p$. If $\text{dex}(S) \subseteq \text{dex}(Q)$, then $\text{dex}(S) = \text{dex}(Q)$.

Proof. Immediate, by induction on the complexity degree of $Q$. \qed

Proposition 7. Let $L, P \subseteq \text{SequDec}$, with $L$ complete, $P$ wf and $L \subseteq P$. Then $L = P$.

Proof. By induction on the structure of $P$, it is an immediate consequence of Proposition 6. \qed

Proposition 8. Let $L \xrightarrow{(a, \phi)} L'$ be obtained from the rules of Table 3. If $L$ is wf then $L'$ is wf.

Proof. By structural induction on $L$, taking into account the different rules that we can apply according to its structure. \qed

Proposition 9. Let $P \subseteq \text{SequDec}$ be wf such that $P \cap \text{Dec} \neq \emptyset$, $L \in P \cap \text{Dec}$, and $L \xrightarrow{(r, \omega)}_p L'$. Then $L' \cup (P \setminus \{L\})$ is wf.

Proof. It is again a straightforward induction on the structure of $P$. \qed

Proposition 10. Let $P \subseteq \text{SequDec}$ be wf such that $P \cap \text{Dec} = \emptyset$, and $L \subseteq P$, with $L \xrightarrow{(a, \phi)} L'$. Then $L' \cup (P \setminus \{L\})$ is wf.

Proof. Similar to the previous one, by induction on the structure of $P$, and using the previous propositions. \qed

With the following property we show that the marked places of a reachable marking constitute a well-formed set of sequential and decision components. Furthermore, the example below the property shows that there can be reachable markings that are not complete.

Proposition 11. Let $Q \in \text{RTPAL}_p$, $N \parallel Q = (P, T, F, \lambda, I, r, M_0)$, and $M \in \{M_0\}$, then $P = \{p \in P \mid \text{Card}(M(p)) > 0\}$ is a well-formed set of sequential and decision components.

Proof. By induction on the length of the pt-sequence $\sigma$ leading to $M$. The base case is just an application of Propositions 4 and 5. For the general case we take $\sigma = M_0[\sigma']M'[R]M$, and the reasoning now depends on $M'$. If $M'$ is stable we just need to apply Proposition 10; otherwise Proposition 9 is applied to conclude the proof. \qed

Example 6. Let us consider the following $\text{RTPAL}_p$ term:

$$N = ((a; i; \text{stop}) \parallel (a; b; i; a(0, \infty); i; \text{stop}) \square (c; i; \text{stop}) \parallel (c; i))$$
Then,

\[ \text{dex}(N) = \{ (a; i; \text{stop}||_{\{a\}} \mathrel{\Box} c; i; \text{stop}||_{\{c\}}), (a; i; \text{stop}||_{\{a\}} \mathrel{\Box} \{c\}; i; d; i; \text{stop}), \\
\quad (\{a\}; b; i; a < 0, \infty; i; \text{stop}) \Box (\{a\}; c; i; \text{stop}) \} \]

From this set of sequential components we can evolve by executing the action \( b \), thus obtaining the following set of sequential and decision places:

\[ \{ (a; i; \text{stop}||_{\{a\}} \mathrel{\Box} c; i; \text{stop}||_{\{c\}}), (a; i; \text{stop}||_{\{a\}} \mathrel{\Box} \{c\}; i; d; i; \text{stop}) \} \]

It is immediate to check that this set is well-formed, but not complete. In Fig. 9 we can see the TPNet of \( N \), annotated with the reachable marking that corresponds to this set of sequential and decision components.

From the preceding properties we can now prove that the MTPNets obtained from Definition 21 are safe.

**Corollary 1.** Let \( Q \in \text{RTPAL}_p \), then \( N || Q || \) is 1-safe.

**Proof.** Let \( M \) be a reachable marking of \( N || Q || \). By Proposition 11 it follows that \( P = \{ p \in P \mid \text{Card}(M(p)) = 0 \} \) is wf. Furthermore, \( P \) is indeed a set (not a multiset), which can be easily proved from Definition 22. Hence \( N || Q || \) is 1-safe. \( \square \)

### 7.2. Transfer lemmas

In this section we show the equivalence between the dynamic behaviour of a \( \text{RTPAL}_p \) term \( Q \) and its corresponding MTPNet \( N || Q || \). For that purpose we introduce the functions \( \text{upd} \) and \( \text{Evolve} \). With \( \text{upd} \) we will obtain from any well-formed set of sequential components a complete set of sequential components. The function \( \text{Evolve} \) will allow us to adjust the
times on a set of sequential components (it captures the effect of time elapsing on the components).

**Definition 23.** We define the partial function $\text{upd} : \mathcal{P}(\text{Sequ}) \rightarrow \mathcal{P}(\text{Sequ})$ which is only defined for well-formed sets of sequential components. When $\mathcal{P}$ is complete, $\text{upd}(\mathcal{P}) = \mathcal{P}$, otherwise:

$$\text{upd}(\mathcal{P}) = \begin{cases} 
\text{upd}(\mathcal{P}_1 \cup \mathcal{P}_2) & \text{if } \mathcal{P} = \mathcal{P}_1 \cup (\mathcal{P}_2 \square \text{dex}(Q)) \\
\text{upd}(\mathcal{P}_1 \cup \mathcal{P}_2) & \text{if } \mathcal{P} = \mathcal{P}_1 \cup (\text{dex}(Q) \square \mathcal{P}_2) \\
\text{upd}(\mathcal{P}_1) \| \mathcal{P}_2) & \text{if } \mathcal{P} = \mathcal{P}_1 \| \mathcal{P}_2 \\
\text{upd}(\mathcal{P}_1) \setminus a & \text{if } \mathcal{P} = \mathcal{P}_1 \setminus a
\end{cases}$$

It can be easily checked (by structural induction) that $\text{upd}(\mathcal{P})$ is indeed complete, for any well-formed set of sequential components $\mathcal{P}$.

**Definition 24.** We define the function $\text{Evolve} : \text{Sequ} \times \mathbb{N} \rightarrow \text{Sequ}$, by:

- $\text{Evolve}(\text{stop}, n) = \text{stop} \quad \forall n \in \mathbb{N}$
- $\text{Evolve}((a; P), n) = (a; P) \quad \forall n \in \mathbb{N}$
- For the timed prefix:

$$\text{Evolve}((a\langle t_1, t_2 \rangle; P), n) = \begin{cases} 
(a\langle t_1 \cdot n, t_2 - n \rangle; P) & \text{if } t_2 \geq n \\
\text{stop} & \text{otherwise}
\end{cases}$$

- $\text{Evolve}(S \| A, n) = \text{Evolve}(S, n) \| A$
- $\text{Evolve}(A\| S, n) = A\| \text{Evolve}(S, n)$
- $\text{Evolve}(S_1 \square S_2, n) = \text{Evolve}(S_1, n) \square \text{Evolve}(S_2, n)$
- $\text{Evolve}(S\setminus a, n) = \text{Evolve}(S, n)\setminus a$

This definition can be easily extended to sets of components:

$$\text{Evolve}((S_1, \ldots, S_r), n_1, \ldots, n_r) = \{\text{Evolve}(S_1, n_1), \ldots, \text{Evolve}(S_r, n_r)\}$$

With this function we will be able to adjust the times of each component in a set of components that correspond to a certain marking on the net, thus capturing the effect of time elapsing on the components.

**Example 7.** Let us consider the term $Q = a\langle 0, 3 \rangle; i; \text{stop}$. Its corresponding $\text{MTPNet}$ is shown in Fig. 10. The initial marking is given by the set of sequential components $\mathcal{P} = \{a\langle 0, 3 \rangle; i; \text{stop}\}$. Then, when one unit of time has elapsed, the net just evolves aging by one unit the marking of that place. The corresponding term is given by the $\text{Evolve}$ function:

$$\text{Evolve}(a\langle 0, 3 \rangle; i; \text{stop}, 1) = \{a\langle 0, 2 \rangle; i; \text{stop}\}$$

We are finally ready to state the transfer lemmas, whose proofs can be found in Appendices B and C.

**Lemma 1 (First transfer lemma).** Let $S$ be a regular, closed and guarded non-deterministic term of $\text{TPAL}_p$, and $N\| S$ its corresponding $\text{MTPNet}$. Given $M$ a stable marking of $N\| S$, 


and \( P = \{P_1, \ldots, P_k\} \) the set of marked places in \( M \), such that \( \text{upd}(\text{Evolve}(P_1, \ldots, P_k, n_1, \ldots, n_k)) = \text{dex}(Q) \), with \( Q \in \text{RTPAL}_p \), being \( n_i \) the age of the token in the place \( P_i \).

Then, given a sequence: \( M[R]_1 M[R_0]_q M[R_1]_q \ldots M[R_n]_q M' \), where \( M_i \) are not stable, and \( M' \) stable, there is a transition:

\[
Q \xrightarrow{(\lambda(R), U)} P \xrightarrow{i} Q'
\]

where \( U \) is a subset of \( \lambda(R) \) whose elements have associated urgent transitions of \( N\|S\| \), \( q = \prod_{i=0}^n q_i \) and \( \text{upd}(\text{Evolve}(P_1', \ldots, P_r', n_1', \ldots, n_r')) = \text{dex}(Q') \), where \( \{P_1', \ldots, P_r'\} \) is the set of sequential components labelling the places marked in \( M' \), and \( n_i' \) is the age of the token in \( P_i' \).

**Lemma 2** (Second transfer lemma). Let \( Q, Q' \in \text{RTPAL}_p \), and \((B, U) \in \text{Bags}(Q)\), such that:

\[
Q \xrightarrow{(B, U)} P \xrightarrow{i} Q'
\]

Given \( S \in \text{RTPAL}_p \), \( M \) a stable marking of \( N\|S\| \), and \( P = \{P_1, \ldots, P_k\} \) the set of marked places in \( M \), such that \( \text{upd}(\text{Evolve}(P_1, \ldots, P_k, n_1, \ldots, n_k)) = \text{dex}(Q) \), being \( n_i \) the age of the token in the place \( P_i \).

Then, there exists a bag of transitions \( R \), with \( \lambda(R) = B \), and a sequence of bags of \( \tau \)'s, \( \sigma = R_0 \ldots R_n \), such that

\[
M[R]_1 M[R_0]_q M[R_1]_q \ldots M[R_n]_q M'
\]

with \( q = \prod_{i=0}^n q_i \), where \( M_i \) are non-stable markings, and \( M' \) is stable. Furthermore, if \( P' = \{P_1', \ldots, P_r'\} \) is the set of sequential components labelling the places marked in \( M' \), we have

\[
\text{upd}(\text{Evolve}(P_1', \ldots, P_r', n_1', \ldots, n_r')) = \text{dex}(Q')
\]

where \( n_i' \) is the age of the token in \( P_i' \).

**Corollary 2.** Let \( S \in \text{RTPAL}_p \), and \( N\|S\| \) its corresponding MTPNet. Let \( M \) be a stable marking of \( N\|S\| \), and \( P = \{P_1, \ldots, P_k\} \) the set of marked places in \( M \), such that \( \text{upd}(\text{Evolve}(P_1, \ldots, P_k, n_1, \ldots, n_k)) = \text{dex}(Q) \), with \( Q \in \text{RTPAL}_p \) and \( n_i \) the age of the token in \( P_i \).

Then:

(a) In the conditions of the first transfer lemma, if \( R \) is \( u \)-maximal in \( N\|S\| \) at the marking \( M \), then \( (\lambda(R), U) \in \text{Now}(Q) \).
(b) In the conditions of the second transfer lemma, if \((B, U) \in \text{Now}(Q)\), then \(R\) is \(u\)-maximal in \(N\parallel S\) at the marking \(M\).

Proof.
(a) Let us suppose that \((\lambda(R), U) \notin \text{Now}(Q)\), then there is \((B', U') \in \text{Bags}(Q)\) with \(\text{Card}(U') \geq \text{Card}(U)\). By the second transfer lemma, \(\exists R'\), an enabled bag of transitions at \(M\), with a greater number of urgent transitions than \(R\), which is a contradiction since \(R\) is \(u\)-maximal in \(M\).

(b) Let us suppose that \(R\) is not \(u\)-maximal in \(M\), then there exists \(R'\), an enabled bag of transitions at \(M\) with a greater number of urgent transitions than \(R\). Thus, by the first transfer lemma there exists \((B', U') \in \text{Bags}(Q)\) with \(\text{Card}(U') \geq \text{Card}(U)\); hence, the urgency degree of \(Q\) is greater than \(\text{Card}(U)\), which is a contradiction, since \((B, U) \in \text{Now}(Q)\).

\[\square\]

8. Conclusions

We have presented the language \(TPAL_p\), and a translation of a subclass of \(TPAL_p\) terms into a probabilistic version of timed-arc Petri nets (\(TPNets\)). With this language we can describe the behaviour of concurrent systems not only at a functional level, but also considering some quantitative information (time and probabilities). Then, there is a wide range of systems whose behaviour can be described with this language, like real-time systems and fault-tolerant systems. Actually, the main aspects of these systems are captured by the different operators of the language. For instance, in real-time systems it becomes crucial to perform some activities in a bounded period of time (time-outs), which can be described in \(TPAL_p\) by the timed prefix operator. However, the timed prefix operator does not impose the execution of the corresponding action when its time-out is going to expire, and thus the activities described with it can be considered as \emph{soft-tasks} in the terminology of real-time systems. But, of course, in real-time systems we have some tasks or activities whose execution cannot be delayed, and they must be executed immediately, once they are available (\emph{hard-tasks}), or at least its execution must be performed within a certain time. These activities can be described in \(TPAL_p\) by means of the urgency operator and the derived operator \(a(n); P\) that we defined in Section 3.2. In fact, this language has a particular way to deal with urgency, as it requires at any instant the execution of a maximal number or urgent actions. Then, the classical requirement of urgency, in the sense that no time can elapse until the urgent actions are performed is satisfied, but also, as \(TPAL_p\) is based on a step semantics, it enforces the execution of as many urgent actions as possible.

Probabilities are also an important element in the description of real-time and fault-tolerant systems, for instance, we can have an estimation of the probability for a channel to fail in a distributed system, or the probability of some incoming data to belong to a certain type, etc. The probabilistic choice operator of \(TPAL_p\) allows us to describe these probabilistic behaviours, which do not depend on the environment, and are resolved by the system itself attending to the probabilities of each alternative. Thus, \(TPAL_p\) has two types of terms, probabilistic and non-deterministic, and they are clearly separated in the syntax.
Another contribution of this paper is the definition of the Net model that we use to translate the \textit{TPAL}$_p$ specifications. This is a probabilistic extension of timed-arc Petri nets, which essentially introduces a new class of places, the so-called \textit{decision places}, and a new class of transitions (\textit{curgent transitions}). Nevertheless, even although we have defined the Net model specifically to translate the \textit{TPAL}$_p$ specifications, this model has enough features to be used isocantly to describe the behaviour of concurrent systems with timed and probabilistic aspects. Actually, the nets that we obtain from the translation are just a particular class of this model (they are 1-safe).

Then, from a methodological point of view, the specifications written in \textit{TPAL}$_p$ can be translated into an equivalent \textit{TPNet}, which offers us a graphical vision of the system, and what is more important, the possibility to simulate and analyse the system behaviour. In this sense, since we have considered a discrete time model, a reduced reachability graph for bounded \textit{TPNets} can be defined, following similar ideas to those used in [32], and it can be used for the analysis of properties. The problem is, of course, the state explosion problem, i.e., these graphs are still quite large, and thus, some research must be made to apply some of the existing techniques of reduction to them.

\textbf{Appendix A.}

\textbf{Proof of Proposition 1.} By induction on the structure of \( N \), considering all the possible transitions, and the definition of \( \text{Bags}(N) \). For the base case we must consider either \( N = a\langle t_1, t_2 \rangle; P \) or \( N = a; P \) (\( N \) cannot be \textit{stop} since there is a transition \( N \stackrel{(B,U)}{\rightarrow} P \)). Both cases are trivial, because by the rules of table 1 we have \( B = (\{1a\}, \emptyset) \) and \( B = (\{1a\}, \{1a\}) \) respectively.

For the general case, the only operators requiring some additional explanations are the parallel and the hiding operators, because both the external choice and the recursion operators are immediate applications of the induction hypothesis.

For the parallel operator \( (N = N_1 \parallel_A N_2) \) we must take into account that \( N \stackrel{(B,U)}{\rightarrow} P \) may come from rules N4a, N4b or N4c. If it comes from N4a (N4b is symmetric), it follows that \( P = P_1 \parallel_A i; \text{age}_1(N_2) \), and \( N_1 \stackrel{(B,U)}{\rightarrow} P_1 \), with \( B \downarrow A = \emptyset \). We can apply the induction hypothesis for \( N_1 \) to obtain a transition \( N_1 \stackrel{(B', B' \cap U')}{\rightarrow} P_1' \). Then, we can use rule N4a to obtain

\[
N_1 \parallel_A N_2 \stackrel{(B', B' \cap U')}{\rightarrow} P_1' \parallel_A i; \text{age}_1(N_2).
\]

When \( N \stackrel{(B,U)}{\rightarrow} P \) comes from rule N4c, it follows that \( P = P_i \parallel_A P_2 \), and \( (B, U) = (B_1 + A B_2, (U_1 \cup A B_2 + U_2 \cup A B_1) \cup A (B_1 + A B_2)) \), \( N_i \stackrel{(B_i, U_i)}{\rightarrow} P_i \), \( i = 1, 2 \), and \( B_1 \downarrow A = B_2 \downarrow A \).

Now we must consider two cases with respect to \( B' \):

(i) If either \( B' \cap B_1 = \emptyset \) or \( B' \cap B_2 = \emptyset \) (\( B' \) only contains actions from one side, let us say \( B_1 \)). In this case we have \( B' \downarrow A = \emptyset \). Then, we apply the induction hypothesis for \( N_1 \) and \( B' \) to obtain a transition \( N_1 \stackrel{(B', B' \cap U_1)}{\rightarrow} P_1' \). Now we can use rule N4a to...
get $N_1 \parallel_A N_2 \xrightarrow{(B', B' \cap U_1)} P'_1 \parallel_A i$; $age_1(N_2)$. Finally, it is easy to check that in this case $B' \cap U = B' \cap U_1$.

(ii) If $B' \cap B_1 \neq \emptyset \land B' \cap B_2 \neq \emptyset$, then we can find $B'_1 \subseteq B_1, B'_2 \subseteq B_2$ such that:

- $B' = B'_1 +_A B'_2$.
- $B' \cap U = (U'_2 \triangleleft_A B'_1 + U'_1 \triangleleft_A B'_2) \triangleleft_A B'$, where $U'_1 = B'_1 \cap U_1$ and $U'_2 = B'_2 \cap U_2$.

Next we show a possible definition of $B'_1, B'_2$ fulfilling both conditions, although we omit the proof, since it is large and tedious. We must distinguish the following cases for that:

1. For $x \in A$ we take $B'_1(x) = B'_2(x) = B'(x)$.
2. For $x \notin A$:
   
   a. If $B'(x) \subseteq U_1(x)$ we take $B'_1(x) = B'(x)$ and $B'_2(x) = 0$.
   
   b. Otherwise, if $B'(x) \subseteq U_2(x)$ we take $B'_2(x) = B'(x)$ and $B'_1(x) = 0$.
   
   c. Otherwise, if $U_1(x) \subseteq B'(x), U_2(x) \subseteq B'(x)$, and $U_1(x) + U_2(x) \supseteq B'(x)$, we take $B'_1(x) = U_1(x)$, and $B'_2(x) = B'(x) - B'_1(x)$.
   
   d. Otherwise: $B'_1(x) = \text{Min}\{B_1(x), B'(x) - U_2(x)\}$, and $B'_2(x) = B'(x) - B'_1(x)$.

Let us suppose that $B'_1 \neq \emptyset, B'_2 \neq \emptyset$. We can apply the induction hypothesis to obtain $N_1 \parallel_A N_2 \xrightarrow{(B'_1, U'_1)} P'_1$. Now, since $B'_1 \downarrow A = B'_2 \downarrow A$, we can apply rule N4c to obtain $N_1 \parallel_A N_2 \xrightarrow{(B_1, B'_1 + A B'_2, U = (U'_1 \triangleleft_A B'_2 + U'_2 \triangleleft_A B'_1) \triangleleft_A (B'_1 + A B'_2))}$, hence $B = B'$ and $U = B' \cap U$.

It may happen that $B'_1 = \emptyset$ (or $B'_2 = \emptyset$), but in this case $B' \downarrow A = \emptyset$ and $B' = B'_1$ (resp. $B' = B'_2$). Then, we can apply the induction hypothesis for $N_2$ and $B'$ obtaining $N_2 \parallel_A N_2 \xrightarrow{(B'_1, B' \cap U_2)} P'_2$, and thus, by rule N4b we have $N_1 \parallel_A N_2 \xrightarrow{(B'_1, B' \cap U_2)} i$; $age_1(N_1) \parallel_A P'_2$. Finally, it is again easy to see that $B' \cap U = B' \cap U_2$, because $B'(x) \subseteq U_2(x)$ (we are in the case b).

For the hiding operator, given a transition $N \backslash a \xrightarrow{(B, U)} P \backslash a$, we must have $(B, U) = (B_a[\tau/a], U_a[\tau/a])$, where $(B_a, U_a) \in Bags(N)$, and a transition $N \xrightarrow{(B_a, U_a)} P$. Considering:

$$B_a(a) = n, \quad B_a(\tau) = m \Rightarrow B(\tau) = n + m$$
$$U_a(a) = r, \quad U_a(\tau) = s \Rightarrow U(\tau) = r + s$$
$$B'(\tau) = t$$

we define

$$B'_a(x) = \begin{cases} B'(x) & \text{if } x \neq a, x \neq \tau \\ t & \text{if } t < r, x = a \\ 0 & \text{if } t < r, x = \tau \\ r & \text{if } t \leq r + s, t \geq r, x = a \\ t - r & \text{if } t \leq r + s, t \geq r, x = \tau \\ r + \text{Min}\{n - r, t - r - s\} & \text{if } t > r + s, x = a \\ t - r - \text{Min}\{n - r, t - r - s\} & \text{if } t > r + s, x = \tau \end{cases}$$
Then, we have:

- $B'_a[t/a] = B'$
- $B'_a \subseteq B_a$
- $(B'_a \cap U_a)[t/a] = (B' \cap U)\)

We omit the proof of these facts, because it requires the distinction of several cases, which are all immediate.

Now we can apply the induction hypothesis for $B'_a$, to obtain a transition $N \xrightarrow{(B'_a \cap U_a)} P'$, which can extended to a transition $N \xrightarrow{(B' \cap U)} P'$, using the rule N6 and the previous facts. □

**Proof of Proposition 2.** Again by induction on the structure of $N$, taking into account the previous proposition. The only case requiring some explanations is the parallel operator, when $(B, U) = (B_1 +_A B_2, (U_1 \lhd_A B_2 + U_2 \lhd_A B_1) \lhd_A (B_1 +_A B_2))$. In this case there will be two transitions (induction hypothesis):

$$N_i \xrightarrow{(B_i, U_i)} P_i \quad i = 1, 2$$

Then, we consider $\tilde{B}_i = B_i \lhd_A (B_1 +_A B_2)$, and $\tilde{U}_i = \tilde{B}_i \cap U_i, i = 1, 2$. It is easy to check that:

- $\tilde{B}_1 +_A \tilde{B}_2 = B$
- $(\tilde{U}_1 \lhd_A \tilde{B}_2 + \tilde{U}_2 \lhd_A \tilde{B}_1) \lhd_A (\tilde{B}_1 +_A \tilde{B}_2) = U$

Using the previous proposition we have two transitions $N_i \xrightarrow{(\tilde{B}_i, \tilde{U}_i)} \tilde{P}_i, i = 1, 2$, and we can apply rule N4c to obtain the transition $N \xrightarrow{(B, U)} \tilde{P}_1 \parallel_A \tilde{P}_2$. □

**Appendix B.**

We need the following proposition for the proof of the first transfer lemma.

**Proposition B1.** Let $\mathbb{P}$ be a well-formed set of sequential and decision components. Then, $\exists \mathbb{P}' \subseteq \mathbb{P}$ such that $\mathbb{P}' \xrightarrow{(a, \phi)} \mathcal{R}$ if and only if $\exists \mathbb{R}' \subseteq \text{upd}(\mathbb{P})$ such that $\mathbb{R}' \xrightarrow{(a, \phi)} \mathcal{R}$.

**Proof.** We proceed by induction on the complexity degree of $Q \in \text{RTPAL}_p$ such that $\text{dex}(Q) = \text{upd}(\mathbb{P})$:

- For \{stop\}, \{a; P\} and \{a\{t_1, t_2\}; P\}: trivial.
- General case: the reasoning is similar for the remaining operators, so we only include the proof for the external choice.

Let $Q = Q_1 \square Q_2$. We have $\text{upd}(\mathbb{P}) = \text{dex}(Q_1) \square \text{dex}(Q_2)$, and by definition of $\text{upd}$ it follows that:

$$\mathbb{P} = \bigcup_{(i, j) \in I \times J} D_i \square D'_j \square \text{dex}(S_{i j}^1) \square \ldots \square \text{dex}(S_{i j}^{k_{ij}}),$$

with $k_{ij} \geq 0$, $\text{dex}(Q_1) = \bigcup_{i \in I} D_i$, and $\text{dex}(Q_2) = \bigcup_{j \in J} D'_j$. 
If we now have \( \mathbb{P}' \xrightarrow{(a, \phi)} \mathbb{R} \), for \( \mathbb{P}' \subseteq \mathbb{P} \), there exists \( I' \subseteq I \) and \( J' \subseteq J \), such that
\[
\bigcup_{(i, j) \in I' \times J'} D_i \square D_j' \square \text{dex}(S_{ij}^1) \square \ldots \square \text{dex}(S_{ij}^k) \xrightarrow{(a, \phi)} \mathbb{R}.
\]

Then:
- If \( \exists i, j \) such that \( D_i \square D_j' \) is complete, it follows from Proposition 6 that \( \text{dex}(Q_1) = D_i \) and \( \text{dex}(Q_2) = D_j' \), hence, \( \text{upd}(\mathbb{P}) = D_i \square D_j' \), and by definition of \( \text{upd} \) we must have \( \mathbb{P} = D_i \square D_j' = \text{dex}(Q_1) \square \text{dex}(Q_2) \).
- If \( D_i \square D_j' \) is not complete, \( \forall i, j \); then, from rule (5) in Table 3, we must have a transition \( \bigcup_{(i, j) \in I' \times J'} D_i \square D_j' \xrightarrow{(a, \phi)} \mathbb{R} \).
- The converse is a straightforward application of rule (5) in Table 3.

**Proof of Lemma 1.** By induction on the degree of complexity of \( Q \in \text{RTPAL}_p \).

**Induction basis:**
- \( Q = \{ \text{stop} \} \). Then, we have \( \text{dex}(Q) = \{ \text{stop} \} = \text{upd}(\text{Evolve}(\mathbb{P}, n)) \), whereby \( \mathbb{P} = \{ \text{stop} \} \) or \( \mathbb{P} = \{ a(t_1, t_2); P \} \), with \( n > t_2 \). In consequence, only time elapsing is possible in the net, to increase the age of the token in \( \mathbb{P} \). Besides, applying Rule N1 of Table 1 we have
\[
\{ \text{stop} \} \xrightarrow{i} \{ \text{stop} \}
\]
with \( \text{upd}(\text{Evolve}(\{ \text{stop} \}, n + 1)) = \{ \text{stop} \} = \text{dex}(\{ \text{stop} \}) \).
- \( Q = \{ a; P \} \). We have \( \text{dex}(Q) = \{ a; P \} = \text{upd}(\text{Evolve}(\{ a; P \}, n_1)) \). Consequently, the only enabled bag is \( R = \{ t_a \} \), with \( t_a = (\{ a; P \}, a, \text{adv}(P)) \). Therefore, given a sequence:
\[
M[\{ t_a \}]_1 M_0[R_0]_{q_0} M_1 \ldots M_n[R_n]_{q_n} M'
\]
where \( M \) is the marking such that there is only one token in the place \{ a; P \} and the remaining places are unmarked, \( M' \) is stable and \( M_i \) are non-stable markings. Furthermore, by rule N2b of Table 1 we have \( Q \xrightarrow{\{a(t_1, t_2)\}} P \).

The proof now depends on the structure of \( P \), we have to show that there exists \( P \overset{i}{\rightarrow}_q Q' \), verifying:
- \( q = \prod_{i=0}^n q_i \)
- \( \text{dex}(Q') = \text{upd}(\text{Evolve}(P'_1, \ldots, P'_k, n'_1, \ldots, n'_{k'})) \), where \( \mathbb{P}' = \{ P'_1, \ldots, P'_k \} \) are the marked places at \( M' \), and \( n'_i \) is the age of the token in \( P'_i \), for \( i = 1, \ldots, k' \).

Reasoning by induction on the structure of \( P \in \text{PPROC} \) we have:
- **Base case:** \( P = i; Q' \). Immediate, since \( \text{adv}(P) = \text{dex}(Q') \), hence we have the sequence \( M[\{ t_a \}]_1 M', \) and thus \( P \overset{i}{\rightarrow}_1 Q' \).
- **For the general case** we have to distinguish the following cases:
  (a) \( P = P_1 \parallel_A P_2 \). Then, \( \text{adv}(P) = \text{adv}(P_1) \parallel_A \text{adv}(P_2) \). We may then split each \( R_i \) into two components, \( R_i^{\text{left}}, R_i^{\text{right}} \), respectively corresponding to transitions with preconditions in \( \text{Dec}_A \) and \( A \parallel \text{Dec} \). Using rule (2) of Table 4 we can divide the sequence \( M_0[R_0]_{q_0} M_1 \ldots M_n[R_n]_{q_n} M' \) into two sequences, one corresponding to...
the firing of $R^\text{left}_{1} \ldots R^\text{left}_{n}$ in $N[P_1]$, and the other one corresponding to the firing of $R^\text{right}_{1} \ldots R^\text{right}_{n}$ in $N[P_2]$.

The proof ends by applying the induction hypothesis to obtain the transitions $P_1 \xrightarrow{i} q'_1$ $Q'_1$ and $P_2 \xrightarrow{i} q'_2$ $Q'_2$, and rule (P3) of Table 2 to conclude that there exists a transition $P_1 \parallel A P_2 \xrightarrow{i} q'_1 q'_2$ $Q'_1 \parallel A Q'_2$, with $q = q'_1$, $q'_2$, and fulfilling the required conditions.

(b) $P = P_1 \setminus b$. This case is a straightforward application of the definition of $\text{adv}$, rule (3) of Table 4, the induction hypothesis and rule (P4) of Table 2.

(c) $P = P_1 + q_0 P_2$. In this case $\text{adv}(P) = \{\tau; \text{adv}(P_1) + q_0 \tau; \text{adv}(P_2)\}$, and we only have two possibilities for $R_0$, either $R_0 = \{(\text{adv}(P), \tau, \text{adv}(P_1))\}$, or $R_0 = \{(\text{adv}(P), \tau, \text{adv}(P_2))\}$. The reasoning is similar for both cases, so we can suppose that we are in the first case. Then, by the induction hypothesis we obtain a transition $P_1 \xrightarrow{i} q_1 q_{n_1} Q'$, and thus, applying rule P2a (or P2b for the other case) of Table 2 we conclude that $P \xrightarrow{i} q_0 q_1 q_{n_1} Q'$.

- $Q = \{a(t_1, t_2); P\}$. In this case we have $\text{dex}(Q) = \{a(t_1, t_2); P\} = \text{upd}(\text{Evolve}(\{a(t'_1, t'_2); P\}, n_1))$, and from this point the reasoning is very similar to the preceding case.

**Induction step:**

- $Q = Q_1 \square Q_2$. For this case we have $\text{upd}(\text{Evolve}(P_1, \ldots, P_k, n_1, \ldots, n_k)) = \text{dex}(Q_1) \square \text{dex}(Q_2)$. Therefore, by definition of $\text{upd}$:

$$\text{Evolve}(P_1, \ldots, P_k, n_1, \ldots, n_k) = \bigcup_{i,j} D_i \square D_j \square \text{dex}(S_{ij}^1) \square \ldots \square \text{dex}(S_{ij}^{k_{ij}}),$$

where $k_{ij} \geq 0$. $\text{dex}(Q_1) = \bigcup_{i \in I} D_i$ and $\text{dex}(Q_2) = \bigcup_{j \in J} D_j$.

Taking now the bag of transitions $R$ and the set of precondition places of its transitions we will obtain a subset of the places marked at $M$. Therefore, we can apply Proposition B1 for each transition $t_k \in R$ to conclude that there must exist $D_{t_k} \subseteq \text{dex}(Q)$ from which $\text{upd}(R)$ can be executed:

$$D_{t_k} \xrightarrow{(\lambda(t_k), \phi_k)} \{t_k^*\}$$

In fact, since $N[S]$ is 1-safe we must have that $D_{t_k} \cap D_{t_{k'}} = \emptyset$, $\forall k, k'$, $k \neq k'$.

Now, by Rule (5) of Table 3 and Proposition 6, and taking into account the structure of $D_{t_k}$ we can conclude that all transitions in $R$ must be executed either by components of $\text{dex}(Q_1)$ or by components of $\text{dex}(Q_2)$. Then, using the induction hypothesis and rule (N3a) of Table 1, we will obtain:

$$Q_1 \square Q_2 \xrightarrow{(\lambda(R), U)} P \xrightarrow{i} q'$$

fulfilling the required conditions.

- $Q = Q_1 \parallel A Q_2$. Then, $\text{upd}(\text{Evolve}(P_1, \ldots, P_k, n_1, \ldots, n_k)) = \text{dex}(Q_1) \parallel A \parallel A \text{dex}(Q_2)$. Consequently, $\text{upd}(\text{Evolve}(P_1, \ldots, P_k, n_1, \ldots, n_k)) = (\bigcup_{i \in I} D_i \parallel A) \cup (\bigcup_{j \in J} A \parallel D_j')$, where $\text{dex}(Q_1) = \bigcup_{i \in I} D_i$ and $\text{dex}(Q_2) = \bigcup_{j \in J} D_j'$. 


Therefore, by definition of \( \text{upd} \) it follows that:

\[
\text{Evolve}(P_1, \ldots, P_k, n_1, \ldots, n_k) = \\
\left( \bigcup_{i \in I} (D_i \circ \text{dex}(S_i^j) \circ \ldots \circ \text{dex}(S_i^{k_i})) \circ A \circ \text{dex}(S_i^{k_i}) \right) \\
\bigcup \left( \bigcup_{j \in J} (A \circ (D_j \circ \text{dex}(T_j^j)) \circ \ldots \circ \text{dex}(T_j^{k_j})) \circ A \circ \text{dex}(T_j^{k_j}) \right)
\]

where \( k_i, k_{1i}, k_j, k_{1j} \geq 0, \forall i, j \).

Taking now \( \hat{R} = \{ t_1, \ldots, t_l \} \), the enabled bag of transitions at \( M \), and its preconditions, we will obtain a subset of the places marked at \( M \). Using Proposition B1 for each \( t_k \in R \) we can conclude that there exists \( D_{t_k} \subseteq \text{dex}(Q) \), from which \( \hat{\lambda}(t_k) \) can be executed:

\[
D_{t_k} \xrightarrow{(\hat{\lambda}(t_k), \phi_k)} \{ t_k^* \}
\]

In fact, since \( N[Q] \) is 1-safe, we must have that \( D_{t_k} \cap D_{t_k'} = \emptyset, \forall k, k', k \neq k' \).

Now we can divide the transitions into \( R \) in three groups:

\[ \begin{align*}
R_{\text{left}} &= \{ t_k \in R \mid \hat{\lambda}(t_k) \notin A, \exists L \parallel A \in D_{t_k} \text{ such that } \{ L \} \xrightarrow{(\hat{\lambda}(t_k), \phi_k)} P, \\
&\quad \text{with } \parallel A = \{ t_k^* \} \\
R_{\text{right}} &= \{ t_k \in R \mid \hat{\lambda}(t_k) \notin A, \exists A \parallel L \in D_{t_k} \text{ such that } \{ L \} \xrightarrow{(\hat{\lambda}(t_k), \phi_k)} P, \\
&\quad \text{with } A \parallel P = \{ t_k^* \} \\
R_{\text{center}} &= \{ t_k \in R \mid \hat{\lambda}(t_k) \in A, \exists L \parallel A, A \parallel L' \in D_{t_k} \text{ such that } \\
&\quad \{ L \} \xrightarrow{(\hat{\lambda}(t_k), \phi_k)} P_1, \{ L' \} \xrightarrow{(\hat{\lambda}(t_k), \phi_k)} P_2, \text{ with } \parallel A \cup A \parallel P_1 = \{ t_k^* \} \}
\end{align*} \]

Then, we consider the sets \( R_1 = R_{\text{left}} \cup R_{\text{center}} \) and \( R_2 = R_{\text{right}} \cup R_{\text{center}} \), and we apply the induction hypothesis for the corresponding pt-sequences on \( N[Q_1] \) and \( N[Q_2] \), to obtain:

\[
Q_1 \xrightarrow{(\hat{\lambda}(R_1), U_1)} P_1 \xrightarrow{i} q_1' \text{ and } Q_2 \xrightarrow{(\hat{\lambda}(R_2), U_2)} P_2 \xrightarrow{i} q_2',
\]

fulfilling the conditions. Finally, we just need to apply rule (N4c) of Table 1 and rule (P3) of Table 2 to conclude that there exists:

\[
Q_1 \parallel A Q_2 \xrightarrow{(B, U)} P_1 \parallel A P_2 \xrightarrow{i} q_1' \parallel A Q_2',
\]

with \( B = \hat{\lambda}(R_1) + A \hat{\lambda}(R_2), U = (U_1 \triangleleft A \hat{\lambda}(R_2) + U_2 \triangleleft A \hat{\lambda}(R_1)) \triangleleft A (\hat{\lambda}(R_1) + A \hat{\lambda}(R_2)), \) and \( q = q_1' \cdot q_2' \). It is now immediate to check that \( \hat{\lambda}(R) = B \), and that \( \text{dex}(Q_1' \parallel A Q_2') \) fulfills the required conditions.

- \( Q = Q_1 \setminus b \). This case is straightforward and the arguments are similar to those followed in the preceding cases, so we omit the proof.
- \( Q = \mu X . Q_1 \). We have \( \text{dex}(Q) = \text{dex}(Q_1(\mu X . Q_1 / X)) \), and:

\[
\text{upd(Evolve}(P_1, \ldots, P_k, n_1, \ldots, n_k)) = \text{dex}(Q) = \text{dex}(Q_1(\mu X . Q_1 / X))
\]
Since \( Q \) is guarded, regular and closed, the structure of \( Q_1 \) allows us to apply the induction hypothesis on \( Q_1 \), and conclude the result. \( \square \)

Appendix C.

We need the following property to prove the second transfer lemma.

**Proposition C1.** Let \( S, Q, Q' \in RTPAL_p \), and \( (B, U) \in Bags(Q) \) with:

\[
Q \xrightarrow{(B,U)} P \xrightarrow{i} q Q'
\]

Let \( M \) be a stable marking of \( N[S] \), \( q = \{ P_1, \ldots, P_k \} \) the set of marked places at \( M \), such that \( Evolve(P_1, \ldots, P_k, n_1, \ldots, n_k) = dex(Q) \), where \( n_i \) is the age of the token on \( P_i \). Then, there exists \( R \), a bag of transitions of \( N[S] \), with \( \lambda(R) = B \), and a sequence \( \sigma = R_0, \ldots, R_n \), where \( R_i \) is a bag of \( \tau \)'s for \( i = 0, \ldots, n \); such that:

\[
M[R]_1,0 M_0[R_0]_{q_0} M_1[\ldots M_n[R_n]_{q_n} M'
\]

with \( q = \prod_{i=0}^n q_i, M_i \) a non-stable marking \((\forall i = 0, \ldots, n)\), and \( M' \) a stable marking such that \( \text{upd}(\text{Evolve}(P'_1, \ldots, P'_k, n'_1, \ldots, n'_k)) = dex(Q') \), where \( q = \{ P'_1, \ldots, P'_k \} \) is the set of sequential components labelling the places marked at \( M' \), and \( n'_{i} \) is the age of the token on \( P'_i \).

**Proof.** By induction on the structure of \( Q \).

**Induction basis:**
- \( Q = stop \). Immediate.
- \( Q = a; P \). In this case we must have \( P = \{ a; P \} \), and \( Evolve(a; P, n_1) = dex(\{ a; P \}) = \{ a; P \} \). Now, we take \( R = \{ t \} \) with \( t = (a; P, a, adv(P)) \). Then, we obtain \( M[\{ t \}]_1,0 M_0 \), where \( M_0 \) corresponds to \( adv(P) \). The reasoning now depends on the structure of \( P \), then we proceed by induction on this structure:

**Induction basis:**
- \( P = \tau; Q'. Immediate.

**Induction step:**
- \( P = S_1 ||_A S_2 \). Then \( adv(P) = adv(S_1)||_A \cup_1 ||_A adv(S_2) \). By rule P3 of Table 2 there exist \( S_1 \xrightarrow{i_1} r_1 Q'_1 \) and \( S_2 \xrightarrow{i_2} r_2 Q'_2 \) with \( q = r_1, r_2 \) and \( Q' = Q'_1 ||_A Q'_2 \). Taking now \( N[S'_1] \) and \( N[S'_2] \), with initial markings \( M'_0 \) and \( M'_2 \), respectively corresponding to \( adv(S_1) \) and \( adv(S_2) \), and applying the induction hypothesis we obtain two sequences:

\[
M'_0[R'_0]_{q'_0} M'_1[R'_1]_{q'_1} \ldots M'_2[R'_2]_{q'_2} M'
\]
frequenting the required conditions. Then, we just need to apply rule (2) of Table 4 to conclude that the corresponding transitions can also be executed on $N[S]$, and the previous sequences can be joined in some different ways to generate a sequence $R_0 \ldots R_n$ fulfilling the conditions.

- $P = S_1 \setminus b$. This case is immediate, because the hiding of $b$ does not affect the probabilistic transitions, so it is an immediate application of the induction hypothesis.

- $P = P_1^0 + q_0 P_2^0$. In this case $adv(P) = \{\tau; adv(P_1^0) + q_0 \tau; adv(P_2^0)\}$, and, by rules P2a and P2b of Table 2, we have either $P_1^0 \xrightarrow{i} q/q_0 Q'$ or $P_2^0 \xrightarrow{i} q/q_0 Q'$. The reasoning is similar for both cases, so let us suppose that we are in the first case. Then, using the induction hypothesis we obtain a sequence:

$$M_1[R_1]_{q_1} \ldots M_n[R_n]_{q_n} M'$$

where $q_1 \ldots q_n = q/q_0$, and $M_1$ is the marking that corresponds to $adv(P_1^0)$. Hence, taking $M_0$ as the marking that corresponds to $adv(P)$ we obtain the sequence:

$$M_0[R_0]_{q_0} M_1[R_1]_{q_1} \ldots M_n[R_n]_{q_n} M'$$

where $R_0 = \{adv(P), \tau, adv(P_1^0)\}$.

- $Q = a(t_1, t_2); P$. In this case we have $Bags(Q) = \{(1.a, \emptyset)\}$, $k = 1$, and $Evolve(P_1, n_1) = dex(Q_1) \sqcup dex(Q_2)$, and $Bags(Q) = Bags(Q_1) \cup Bags(Q_2)$. Hence, either $(B, U) \in Bags(Q_1)$ or $(B, U) \in Bags(Q_2)$, we may assume that the first one occurs (the reasoning for the other case is analogous). Then, by rule N3a of Table 1 there exists $Q_1 \xrightarrow{(B,U)} P \xrightarrow{i} q Q'$. We can then apply the induction hypothesis on $N[Q_1]$, to obtain a sequence:

$$M^1[R_1]_{1,0} M_0[R_0]_{q_0} \ldots M_n[R_n]_{q_n} M'$$

fulfilling the required conditions. In fact, this sequence is also a sequence of $N[S]$, which can be proved by using rule (5) of Table 3.

- $Q = Q_1 \sqcup A Q_2$. In this case we have $Evolve(P_1, \ldots, P_k, n_1, \ldots, n_k) = dex(Q_1) \sqcup A |dex(Q_2)$, and $Bags(Q) = Bags(Q_1) \oplus A Bags(Q_2)$, hence, three cases may occur:

  a) $B \downarrow A = \emptyset$ and $(B, U) \in Bags(Q_1)$. By rule N4a of Table 1 there exists $Q_1 \xrightarrow{(B,U)} P_1$, and thus $Q_1 \sqcup A Q_2 \xrightarrow{(B,U)} P_1 \sqcup A Q_1$. Furthermore, by rule P3 of Table 2 there must exist a transition $P_1 \xrightarrow{i} q Q'_1$. Then, we apply the induction hypothesis on $N[Q_1]$, to obtain a sequence:

$$M^1[R_1]_{1,0} M_0[R_0]_{q_0} \ldots M_n[R_n]_{q_n} M'$$

fulfilling the conditions, where the set of marked places in $M^1$ is $dex(Q_1)$. This sequence is also a sequence of $N[S]$, which can be proved by using rule (3) of Table 3 and rule
(2) of Table 4. Then, adapting the markings and the transitions to $N[\top S]$ we obtain the corresponding sequence on this net:

$$M[R]_{1.0}M_0[R_0]_{q_0} \ldots M_n[R_n]_{q_n}M'$$

Finally, notice that if $P'$ is the set of sequential and decision components corresponding to the marking $M'$, then $P' = P'[A] \cup A \| \text{dex}(Q_2)$, where $P'[A]$ is the set of sequential and decision components that corresponds to the marking $M'$ on $N[Q_1]$. Taking $P' = \{ P'_1, \ldots, P'_k \}$, $\text{dex}(Q_2) = \{ L_1, \ldots, L_{k'_2} \}$, and $k' = k'_1 + k'_2$, we have:

$$\text{upd}(\text{Evolve}(P'_1[A], A \| L_1, \ldots, A \| L_{k'_2}, n'_1, \ldots, n'_k))$$

$$= \text{upd}(\text{Evolve}(P'_1, \ldots, P'_k, n'_1, \ldots, n'_k)) \cup A \| \text{upd}(\text{Evolve}(L_1, \ldots, L_{k'_2}, n_{k'_1+1}, \ldots, n_{k'_k}))$$

$$= \text{dex}(Q'_1[A] \cup A \| \text{dex}(\text{age}_1(Q_2)) = \text{dex}(Q'_1[A \| \text{age}_1(Q_2))$$

Only one tick has elapsed with the firing of $R$, and no time elapses with the sequence of bags of $\tau's$, hence $\text{Evolve}(L_1, \ldots, L_{k'_2}, n_{k'_1+1}, \ldots, n_{k'_k}) = \text{dex}(\text{age}_1(Q_2))$, and thus:

$$\text{upd}(\text{Evolve}(P'_1[A], A \| L_1, \ldots, A \| L_{k'_2}, n'_1, \ldots, n'_k))$$

$$= \text{dex}(Q'_1[A] \cup A \| \text{dex}(\text{age}_1(Q_2)) = \text{dex}(Q'_1[A \| \text{age}_1(Q_2))$$

(b) $B \downarrow A = 0$ and $(B, U) \in \text{Bags}(Q_2) \setminus A$

This is the symmetric case of (a).

(c) $(B, U) = (B_1 + A B_2, (U_1 \prec A B_2 + U_2 \prec A B_1) \prec A (B_1 + A B_2))$, with $(B_1, U_1) \in \text{Bags}(Q_1)$, $(B_2, U_2) \in \text{Bags}(Q_2)$ and $B_1 \downarrow A = B_2 \downarrow A$.

Then, using rule N4c of Table 1 there exist $Q_1 \xrightarrow{r_1} i q_1$ and $Q_2 \xrightarrow{r_2} j q_2$, $P_1 \xrightarrow{i} q_1$ and $P_2 \xrightarrow{j} q_2$, with $q = q_1, q_2$. Taking $N[Q_1]$ and $N[Q_2]$ and applying the induction hypothesis we obtain two sequences:

$$M[R]_{1.0}M_0[R_0]_{q_0} \ldots M_n[R_n]_{q_n}M'$$

$$M[R]_{1.0}M_0[R_0]_{q_0} \ldots M_n[R_n]_{q_n}M'$$

where $M'$ corresponds to $\text{dex}(Q_i)$, for $i = 1, 2$, $q_1 = \prod_{i=0}^k q_i$ and $q_2 = \prod_{i=0}^k q_i$, fulfilling the required conditions.

Then, using a similar procedure to that used in the proof of the first transfer lemma for the case of the parallel operator, and applying rules (3) and (4) of Table 3 we can generate a set of transitions $R$ such that $M[R]_{1.0}M_0$

By the induction hypothesis, we can also conclude that the set of marked places in $M_0$ is $\text{adv}(P_1[A] P_2)$. Taking now into account that $M_0$ corresponds to $\text{adv}(P_1)$ and $M_0'$ corresponds to $\text{adv}(P_2)$, it is straightforward to construct a sequence of bags of $\tau's$, $R_0 \ldots R_n$, executable from $M_0$ (using rule (2) of Table 4). This can be made by joining the sequences $R_1 \ldots R_i$ and $R_0 \ldots R_{i'}$, but notice that this can be made in many different ways.
Finally, the proof ends just observing that the obtained marking after performing the sequence $R_0 \ldots R_n$ corresponds to $P' = P^{1'} \parallel A \cup A \parallel P^{2'}$, where $P^{1'}$ corresponds to $M^1$ and $P^{2'}$ to $M^2$, and using the induction hypothesis.

- $Q = Q \setminus b$. This case is straightforward, using similar arguments to those used in the preceding cases.
- $Q = \mu X.N$. We have $\overline{Evolve}(P_1, \ldots, P_k, n_1, \ldots, n_k) = \text{dex}(N(\mu X.N/X))$, $\text{Bags}(\mu X.N) = \text{Bags}(N(\mu X.N/X))$, and by rule N5 of Table 1 there exists $N(\mu X.N/X) \xrightarrow{(B,U)} P$, since $Q \in \text{RTPAL}_p$. Then, the result is immediate by applying the induction hypothesis on $N(\mu X.N/X)$.

**Proof of Lemma 2.** We take $N\parallel Q$ with its initial marking $M^1$ (all tokens have age 0). Then, applying Proposition C1 we obtain a sequence:

$$M^1[R^1]_{1,0}M_0[R_0]q_0M_1[\ldots M_n[R_n]q_0M'$$

with $q = \prod_{i=0}^n q_i$. $M_i$ a non-stable marking ($i = 0, \ldots, n$), and $M'$ a stable marking, fulfilling:

$$\text{upd}(\overline{Evolve}(P'_1, \ldots, P'_k, n'_1, \ldots, n'_k)) = \text{dex}(Q')$$

where $\mathbb{P}' = \{P'_1, \ldots, P'_k\}$ is the set of marked places in $M'$, and $n'_i$ is the age of the token in $P'_i$. Taking now into account that we are starting from a marking $M'$ in which all tokens have age 0, we can conclude that $n'_i \leq 1$, $i = 1, \ldots, k$. In fact, we can split $\mathbb{P}'$ into two parts: $\mathbb{P}' = \mathbb{P}'_1 \cup \mathbb{P}'_2$, where:

$$\mathbb{P}'_1 = \{P \in \mathbb{P}' | P \in t^*, \ t \in R \vee t \in R_i, \ i = 0, \ldots, n\}$$

$$\mathbb{P}'_2 = \mathbb{P}' \setminus \mathbb{P}'_1$$

Then, for all $P \in \mathbb{P}'_1$ it follows that the token on this place must age 0, and the tokens on the places in $\mathbb{P}'_2$ must have age 1. We can now enumerate the elements of $\mathbb{P}'$ as follows: $\mathbb{P}' = \{P'_1, \ldots, P'_{k'_1}, P'_{k'_1+1}, \ldots, P'_{k'}\}$, where $\mathbb{P}'_1 = \{P'_1, \ldots, P'_{k'_1}\}$, and $\mathbb{P}'_2 = \{P'_{k'_1+1}, \ldots, P'_{k'}\}$.

If we now consider each $t_k \in R^1$, we must have $\gamma_k \subseteq \text{dex}(Q)$, and a transition $\gamma_k \xrightarrow{(\lambda(t_k), \phi_k)} t_k^*$. Therefore, by Proposition B1 there exists $\exists P''_k \subseteq \overline{Evolve}(P_1, \ldots, P_k, n_1, \ldots, n_k)$ and a transition $\gamma_k \xrightarrow{\lambda(t_k), \phi_k} t_k^*$. Taking $R = \{(P''_k, \lambda(t_k), t_k^*) | t_k \in R^1\}$, it follows that this set of transitions $R$ can be fired from the marking $M$. Actually, from the marking generated by this firing we can fire in a row the sets of transitions $R_i$, to obtain a sequence:

$$M[R]_{1,0}M'_0[R'_0]q_0M'_1[\ldots M'_n[R'_n]q_0M''$$

such that the set of places marked at $M''$ is $\mathbb{P}'' = \mathbb{P}'_1 \cup \mathbb{P}'_2$, with $\mathbb{P}'_1 = \mathbb{P}'_1$ and $\mathbb{P}'_2 = \{P''_{k'_1+1}, \ldots, P''_{k'}\}$, fulfilling:

$$\text{upd}(\overline{Evolve}(P'_1, \ldots, P''_{k'_1}, P''_{k'_1+1}, \ldots, P''_{k'}, \overline{\overline{0}, \ldots, 0, n''_{k'_1+1}, \ldots, n''_{k'}})) = \text{dex}(Q'')$$

for some $Q''$, where $n''_i$ is the age of the token on $P''_i$, for $i = k'_1 + 1, \ldots, k'$. \[\square\]
Acknowledgements

This work has been partially supported by the CICYT project “Description and Evaluation of Distributed Systems and Application to Multimedia Systems”, TIC2003-07848-C02-02.

References