Verifying properties in multi-agent systems using Stochastic Petri Nets and Propositional Dynamic Logic

Bruno Lopes\textsuperscript{1}, Mario Benevides\textsuperscript{2}, Edward Hermann Haeusler\textsuperscript{1}

\textsuperscript{1}Departamento de Informática  
Pontifícia Universidade Católica do Rio de Janeiro  
Rio de Janeiro, Brazil  
\{bvieira,hermann\}@inf.puc-rio.br

\textsuperscript{2}Programa de Engenharia de Sistemas e Computação  
Universidade Federal do Rio de Janeiro  
Rio de Janeiro, Brazil  
mario@cos.ufrj.br

Abstract. Petri Nets are widely used to model concurrent systems. Multi-agent systems are composed by many independent agents where the work of some may depend on the work of others. In this work we illustrate the usage of a sound, complete and decidable logic system to verify properties in multi-agent systems: the DS\textsuperscript{3} logic that takes advantage of the intuitive graphical interpretation of Petri Nets. We present the Petri Net formalism adopted, the logical system and an usage example.

1. Introduction

Multi-agent systems are composed by several agents that aim to eventually reach some goals that need the contribution of another agents [Khosravifar 2013]. They are widely used in Artificial Intelligence.

Petri Nets are a formalism to specify concurrent systems which has an intuitive graphical interpretation. As agents are independent but may need help of other agents to reach some goal, Petri Nets can offer a background to model such systems. There is a large literature about modelling multi-agent systems using Petri Nets [Khosravifar 2013, Everdij et al. 2006, Holvoet 1995, Perše et al. 2008, Rongier and Liégeois 1999].

Stochastic Petri Nets are an extension of Petri Nets where each transition is associated with a random variable of the exponential distribution. This model of Petri Nets is widely used for non-linear time-modelling and, as ordinary Petri Nets, it has a useful graphical interpretation. The random variables will control the firing rate of each transition. So, a transition will only fire when it is enabled and its timing achieves zero.

Petri-PDL [Benevides et al. 2011] is a sound and complete extension of Propositional Dynamic Logic (PDL) [Fischer and Ladner 1979] that is a multi-modal logic used for reasoning about Petri Net programs where each modality is composed of a program \(\pi\) with a markup \(s\), says \(\langle s, \pi \rangle\). The interpretation of a formula \(\langle s, \pi \rangle \varphi\) is such that after the Petri Net program \(\pi\) runs with initial markup \(s\) there is a state reachable from the initial state where \(\varphi\) holds (and analogous for a “necessity” modality where \(\varphi\) must hold in all states accessible from the initial: \([s, \pi] \varphi \leftrightarrow \neg \langle s, \pi \rangle \neg \varphi\)).
There are some stochastic approaches to dynamic logics for reasoning about concurrent systems, increasing their expressiveness [Feldman 1983, Feldman and Harel 1984, Kozen 1983, Tiomkin and Makowsky 1985, Tiomkin and Makowsky 1991], but all of them lacks on some desirable properties (do not have a finite axiomatization, do not allow boolean combination of propositional variables, are defined only for regular programs, are undecidable, requires a measurable translation of the program or only let one to know if some probability is greater then a constant).

In this work we present and extension of Petri-PDL: the $\mathcal{DS}3$ logic, a sound and complete system (for a defined subset of Stochastic Petri Nets) to deal with Stochastic Petri Nets. In section 2 we present an overview of the Petri Net system used and Petri-PDL. Section 3 presents the $\mathcal{DS}3$ system followed by an usage example in section 4. Section 5 presents the conclusions and further work.

2. Background

In this section we present the Petri Net convention system used in this work, present Petri-PDL syntax and semantics and make a brief review of Stochastic Petri Nets.

2.1. Petri Net convention system

Petri-PDL language uses the Petri Net model defined in the work of de Almeida and Haeusler [de Almeida and Haeusler 1999]. In this model there are only three types of transition which define all valid Petri Nets due to its compositions. These basic Petri Nets are as in figure 1.

![Figure 1. Basic Petri Nets](image)

2.2. Stochastic Petri Nets

A Stochastic Petri Net (SPN) [Marsan and Chiola 1987] is a tuple $\mathcal{P} = \langle S, T, W, M_0, \Lambda \rangle$, where $S$ is a finite set of places, $T$ is a finite set of transitions with $S \cap T = \emptyset$ and $S \cup T \neq \emptyset$ and $W$ is a function which defines directed edges between places and transitions and assigns a $w \in \mathbb{N}$ that represents a multiplicative weight for the transition, as $W : (S \times T) \cup (T \times S) \rightarrow \mathbb{N}$, $M_0$ is the initial markup and $\Lambda = \lambda_1, \lambda_2, \ldots, \lambda_n$ the firing rates of each transition (i.e. the parameter of each exponentially distributed random variable associated with each transition, a way of model the proportion of firing of this transition despite its preset).

In an SPN the firing is determined by the markups and by the firing rate. Each transition $t_i \in T$ is associated to an unique random variable of the exponential distribution with parameters $\lambda_i \in \Lambda$. 
In the initial markup \((M_0)\) each transition sets its firing delay (i.e. a timing for an enabled transition fires) by an occurrence of the random variable whose transition is associated. Each firing delay is marking-dependent and the \(t_i \in T\) firing rate at marking \(M_j\) is defined as \(\lambda_i(M_j)\) and has average firing delay (i.e. the average value of a random variable occurrences) of \(1/\lambda_i(M_j)\). After a firing, each previously non-marking-enabled transition gets a new firing delay by sampling its associated random variable. A previously marking-enabled that keeps marking-enabled has its firing delay decreased in a constant speed. When a transition firing delay reaches zero, this transition fires.

So, given a markup \(M\) of a Petri Net, we say that a transition \(t_i\) is enabled on \(M_j\) if and only if \(\forall x \in t_i, M_j(x) \geq W(x, t_i)\) and \(\lambda_i(M_j) \leq \min(\lambda_1(M_j), \lambda_2(M_j), \ldots, \lambda_n(M_j))\), where \(t_i\) denotes the preset of the transition \(t_i\) and \(t_i^+\) denotes its postset. A new markup generated by setting a transition which is enabled is defined in the same way as in an ordinary Marked Petri Net, i.e.

\[
M_{i+1}(x) = \begin{cases} 
M_i(x) - W(x, t), & \forall x \in t - t^* \\
M_i(x) + W(t, x), & \forall x \in t^* \cap t \\
M_i(x), & \forall x \notin \{t - t^* \cup \{t^* - t\}\}
\end{cases}
\]  

(1)

and a new firing delay for a transition \(t_i\) after a markup \(M_j\) is defined as

\[
\lambda_i(M_{j+1}) = \begin{cases} 
\text{new}_e(\lambda_i) & \text{if} \\{ \begin{array}{l}
\forall x \in t_i, M_j(x) \geq W(x, t_i) \\
\lambda_i(M_j) \leq \min(\lambda_1(M_j), \ldots, \lambda_n(M_j))
\end{array} \\
\text{or} \\{ \begin{array}{l}
\exists x \in t_i, M_j(x) < W(x, t_i) \\
\forall x \in t_i, M_{j+1}(x) \geq W(x, t_i)
\end{array} \\
\text{othercase}
\end{cases}
\]  

(2)

where \(\text{new}_e(\lambda)\) denotes a new occurrence of the random variable exponentially distributed with parameter \(\lambda\) associated to \(t_i\). By default, \(\forall x \forall y W(x, y) = 1\).

### 2.3. Petri-PDL

Petri-PDL [Benevides et al. 2011] is a multi-modal logic for reasoning about Petri Net programs. Its language consists of

<table>
<thead>
<tr>
<th>Propositional symbols:</th>
<th>p, q . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place names: e.g.:</td>
<td>a, b, c, d . . .</td>
</tr>
<tr>
<td>Transition names: e.g.:</td>
<td>(t_1, t_2, t_3 . . .), each transition name has a unique type</td>
</tr>
<tr>
<td>Transition types:</td>
<td>(T_1 : xt_1yz, T_2 : xyt_2z) and (T_3 : xt_3yz), where (x, y) and (z) denote places</td>
</tr>
</tbody>
</table>

| Petri Net Composition symbol: | \(\odot\) |

| Sequence of names: | \(S = \{s, s_1, s_2, . . .\}\), where \(s\) is the empty sequence. We use the notation \(s \prec s'\) to denote that all names occurring in \(s\) also occur in \(s'\). |

**Definition 1. Programs:**

Basic programs: \(\pi_b := at_1b | at_2bc | abt_3c\) where \(t_i\) is of type \(T_i, i = 1, 2, 3\)

Petri Net Programs: \(\pi := \pi_b \mid \pi \odot \pi, the whole composition denoted by \eta_1, \eta_2, . . . .\)

**Definition 2. A formula is defined as (let \(s\) be a sequence of names)**

\[
\varphi := p \mid \top \mid \neg \varphi \mid \varphi \land \varphi \mid (s, \pi)\varphi.
\]
We use the standard abbreviations $\bot \equiv \neg \top$, $\varphi \lor \phi \equiv \neg (\neg \varphi \land \neg \phi)$, $\varphi \rightarrow \phi \equiv \neg (\varphi \land \neg \phi)$ and $[s, \pi] \varphi \equiv \neg (s, \pi) \neg \varphi$.

**Definition 3.** We define the firing function $f : S \times \pi_b \rightarrow S$ as follows

- $f(s, at_1b) = \begin{cases} s_1bs_2, & \text{if } s = s_1as_2 \\ \epsilon, & \text{if } a \not\in s \end{cases}$
- $f(s, abt_2c) = \begin{cases} s_1cs_2s_3, & \text{if } s = s_1as_2bs_3 \\ \epsilon, & \text{if } a, b \not\in s \end{cases}$
- $f(s, at_3bc) = \begin{cases} s_1s_2bc, & \text{if } s = s_1as_2 \\ \epsilon, & \text{if } a \not\in s \end{cases}$
- $f(\epsilon, \eta) = \epsilon, \text{ for all petri nets programs } \eta$.

**Definition 4.** A frame for Petri-PDL is a 3-uple $\langle W, R, M \rangle$, where

- $W$: a non-empty set of states;
- $M : W \rightarrow S$;
- $R$ is a binary relation over $W$, for each basic program $\alpha \prec \pi_b$, satisfying the following condition. Let $s = M(w)$
- if $f(s, \alpha) \neq \epsilon$, $wR_v \text{iff } f(s, \alpha) \prec M(v)$
- if $f(s, \alpha) = \epsilon$, $wR_v \text{iff } w = v$
- we inductively define a binary relation $R_{\eta}$, for each Petri Net program $\eta = \eta_1 \circ \eta_2 \circ \cdots \circ \eta_n$, as $R_{\eta} = \{ (w, v) \mid \text{for some } \eta_i, \exists u \text{ such that } s_i \prec M(u) \text{ and } wR_{\eta_u} u \text{ and } uR_{\eta_v} v \}$
- Where $s_i = f(s, \eta_i)$, for all $1 \leq i \leq n$.

**Definition 5.** A model for Petri-PDL is a pair $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$, where $\mathcal{F}$ is a PDL frame and $\mathcal{V}$ is a valuation function $\mathcal{V} : \Phi \rightarrow ^{2^W}$, where $\Phi$ is the set of all valid propositions.

The semantical notion of satisfaction for Petri-PDL is defined as follows.

**Definition 6.** Let $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$ be a model. The notion of satisfaction of a formula $\varphi$ in a model $\mathcal{M}$ at a state $w$, notation $\mathcal{M}, w \models \varphi$, can be inductively defined as follows:

- $\mathcal{M}, w \models p \iff w \in \mathcal{V}(p)$;
- $\mathcal{M}, w \models \top$ always;
- $\mathcal{M}, w \not\models \neg \varphi \iff \mathcal{M}, w \not\models \varphi$;
- $\mathcal{M}, w \models \varphi_1 \land \varphi_2 \iff \mathcal{M}, w \models \varphi_1$ and $\mathcal{M}, w \models \varphi_2$;
- $\mathcal{M}, w \models \langle s, \eta \rangle \varphi$ if there exists $v \in W$, $wR_{\eta} v$, $s \prec M(w)$ and $\mathcal{M}, v \models \varphi$.

If $\mathcal{M}, v \models A$ for every state $v$, we say that $A$ is valid in the model $\mathcal{M}$, notation $\mathcal{M} \models A$. And if $A$ is valid in all $\mathcal{M}$ we say that $A$ is valid, notation $\models A$.

Its axiomatic system is composed by the following rules.

**(PL)** Enough propositional logic tautologies

**(K)** $[s, \pi] (p \rightarrow q) \rightarrow ([s, \pi] p \rightarrow [s, \pi] q)$

**(Du)** $[s, \pi] p \leftrightarrow [s, \pi] \neg \neg p$

**(PC)** $\langle s, \eta \rangle \varphi \leftrightarrow \langle s, \eta_1 \rangle \langle s_1, \eta \rangle \varphi \lor \langle s, \eta_2 \rangle \langle s_2, \eta \rangle \varphi \lor \cdots \lor \langle s, \eta_n \rangle \langle s_n, \eta \rangle \varphi$

where $s_i = f(s, \eta_i)$, for all $1 \leq i \leq n$.

**(R**$)$ **(s, \eta) \varphi \leftrightarrow \varphi$, if $f(s, \eta) = \epsilon$
(Sub) If $\models \varphi$, then $\models \varphi^\sigma$, where $\sigma$ uniformly substitutes proposition symbols by arbitrary formulas.

(MP) If $\models \varphi$ and $\models \varphi \rightarrow \psi$, then $\models \psi$.

(Gen) If $\models \varphi$, then $\models [s, \pi] \varphi$.

The system was proved to be sound for all Petri Nets and decidable and complete for the set of all Petri Nets which do not have places that accumulate tokens indefinitely (as pointed out by the work of A. Mazurkiewicz [Mazurkiewicz 1987, Mazurkiewicz 1989], logics that deal with Petri Nets use to be undecidable and incomplete by these kind of places) by proving the soundness of the axioms, showing that a filtered model has finitely many states and showing by a canonic model that if a formula is true in all models then there is a derivation of it. The proofs follow the methodology of [Blackburn et al. 2001, Harel et al. 2000] and [Goldblatt 1992] and are in http://www.tecmf.inf.puc-rio.br/BrunoLopes/Proofs.

3. A stochastic approach to Petri-PDL

The $\mathcal{DS}_{3}$ logic (Propositional Dynamic Logic for Stochastic Petri Nets) presents a stochastic approach for Petri-PDL.

The language of $\mathcal{DS}_{3}$ is the same as that of Petri-PDL, differing in its frame definition.

**Definition 7. $\mathcal{DS}_{3}$ Frame**

A frame for $\mathcal{DS}_{3}$ is a 5-uple $F_{3} = \langle W, R_\pi, M, \Pi, \Lambda, \delta \rangle$ where

- $W$ is a non-empty set of states
- $M : W \rightarrow S$
- $\Pi$ is a finite Stochastic Petri Net such that for any program $\pi$ used in a modality, $\pi \in \Pi$ (i.e. $\pi$ is a subnet of $\Pi$)
- $\Lambda(\pi) = \langle \lambda_1, \lambda_2, \ldots, \lambda_n \rangle$ is the sequence of $\mathbb{R}^{+}$ values denoting the fire rate of each transition of $\pi_1 \odot \pi_2 \odot \cdots \odot \pi_n = \pi \in \Pi$
- $\delta(w, \pi) = \langle d_1, d_2, \ldots, d_n \rangle$ is the sequence of firing delays of the program $\pi \in \Pi$ in the world $w \in W$ respectively for each program $\pi_1 \odot \pi_2 \odot \cdots \odot \pi_n = \pi$, satisfying the following conditions (let $s = M(w)$ and $r = M(v)$)
  - if $wR_{\pi_\alpha}v$, $f(v, \pi_b) = \epsilon$, $\delta(w, \pi_b) = \delta(v, \pi_b)$
  - if $f(s, \pi_b) = \epsilon$, $f(r, \pi_b) \neq \epsilon$ and $wR_{\pi_\alpha}v$, $\delta(v, \pi_b)$ is an occurrence of a random variable of exponential distribution with parameter $\Lambda(\pi_b)$
  - if $f(s, \pi_b) \neq \epsilon$, $f(r, \pi_b) \neq \epsilon$ and $wR_{\pi_\alpha}v$, $\delta(v, \pi_b) < \delta(w, \pi_b)$
- $R_\alpha$ is a binary relation over $W$, for each basic program $\alpha \in \pi_\alpha$, satisfying the following conditions (let $s = M(w)$)
  - if $f(s, \alpha) \neq \epsilon$ and $\delta(w, \alpha) = \min(\delta(w, \Pi))$, $wR_{\alpha}v$ iff $f(s, \alpha) < M(v)$
  - if $f(s, \alpha) = \epsilon$ or $\delta(w, \alpha) \neq \min(\delta(w, \Pi))$, $wR_{\alpha}v$ iff $w = v$
- we inductively define a binary relation $R_\eta$ for each Petri Net program $\eta = \eta_1 \odot \eta_2 \odot \cdots \odot \eta_n$, as $R_\eta = \{(w, v) \mid \exists \eta_i, \exists u \text{ such that } s_i \prec M(u) \text{ and } wR_{\eta_i}u \text{ and } \delta(w, \eta_i) = \min(\delta(w, \Pi)) \text{ and } uR_{\eta}v \}$ where $s_i = f(s, \eta_i)$, for all $1 \leq i \leq n$.

**Definition 8. $\mathcal{DS}_{3}$ Model**

A model for $\mathcal{DS}_{3}$ is a pair $\mathcal{M} = \langle F_{3}, V \rangle$, where $F_{3}$ is an $\mathcal{DS}_{3}$ frame and $V$ is a valuation function $V : \Phi \rightarrow 2^W$. 
Definition 9. Semantic notion of DS3

Let $M_3$ be a model for DS3. The notion of satisfaction of a formula $\varphi$ in $M_3$ at a state $w$, says $M_3,w \models \varphi$ is inductively defined as follows.

- $M_3,w \models p$ iff $w \in V(p)$
- $M_3,w \models \top$ always
- $M_3,w \models \neg \varphi$ iff $M_3,w \not\models \varphi$
- $M_3,w \models \varphi_1 \land \varphi_2$ iff $M_3,w \models \varphi_1$ and $M_3,w \models \varphi_2$
- $M_3,w \models (s,\eta)\varphi$ if there exists $v \in W$, $wR_{\eta}v$ and $\Pr(M_3,v \models (s,\eta)\varphi | \delta(v,\Pi)) > 0$ (note that $\Pi$ is the SPN of the model).

So if we say that $M_3,w \models (s,\eta)\varphi$ then it means that the program $\eta$ beginning with the markup $s$ has probability of running greater then one (i.e. the probability of a firing happens is greater then zero) and that when it stops $\varphi$ holds in the current state. If $\varphi$ is valid in all states of $M_3$ then $\varphi$ is valid in $M_3$, says $M_3 \models \varphi$; and if $\varphi$ is valid in any model then $\varphi$ is valid, says $\models \varphi$.

Lemma 1. Truth Probability of a Modality

The probability of $M_3,w \models (s,\pi_b)\varphi$ is (let $s = M(w)$)

$$\Pr(M_3,w \models (s,\pi_b)\varphi | \delta(w,\Pi)) = \sum_{\pi_b \in \Pi : \delta(w,\pi_b) \neq \epsilon} \frac{\delta(w,\pi_b)}{\sum_{\pi_b \in \Pi : \delta(w,\pi_b) \neq \epsilon} \delta(w,\pi_b)}$$

The axiomatic system of DS3 is the same then Petri-PDL.

The system is proved to be sound, decidable and complete for the normalised Stochastic Petri Nets (as for Petri-PDL). The complete proofs are available in http://www.tecmf.inf.puc-rio.br/BrunoLopes/Proofs.

4. Usage example

The Petri Net in figure 2 presents a scenario where four agents ($A_1$, $A_2$, $A_3$ and $A_4$) must collect and process some data from the resource centre ($r$), but agents $A_1$ and $A_2$ cannot make the full process and needs that $A_3$ or $A_4$ completes the computation. Another characteristic of this system is that $A_3$ and $A_4$ have a faster processor then $A_1$ and $A_2$ and that $A_1$ and $A_2$ are in a shared memory system, but the clock of the processor of $A_1$ is faster then $A_2$. As the clock of the processor of $A_1$ is faster then the one of $A_2$, the firing rates (i.e. the $\lambda$ parameter of the random variable which is associated with the transitions whose preset or postset depends on $A_1$) is greater then the ones of $A_2$.

Taking a propositional formula $p$ that means that all data was processed, the formula $\{r\{r\{r\{r\}m\}, rmt_2A_1 \circ rmt_2A_2 \circ rt_1A_3 \circ rt_1A_4 \circ A_1t_3A_3m \circ A_2t_3A_3m \circ A_1t_3A_4m \circ A_4t_3A_4m\}p$ says that after some running of the Petri Net of figure 2, $p$ holds, that is, all the data are processed. Verify if this formula holds in a state $w$ of a model $M$ (i.e. verify if it is possible that some transition fires) is equivalent to compute the probability of some basic program fire is greater then zero, which is reduced to the equation in lemma 1. To verify if it is possible that $A_1$ and $A_2$ compute some data in parallel, we verify that after some of them begin to process something (i.e. $rmt_2A_1$ or $rmt_2A_2$ fires), $m$ will not
be anymore in the sequence of names, so it is not possible that the other agent starts to compute something unless a transition that restates a token to m fires.

If it is desirable to know if, from a state \(w\), it is possible that some agent (e.g. agent \(A_1\)) collect some data to process, it is just needed to compute \(\Pr(M, w \models (s, rmt_2A_1) \sqcup \delta(w, rmt_2A_1 \circ rmt_2A_2 \circ rt_1A_3 \circ rt_1A_4 \circ A_1t_3A_3m \circ A_2t_3A_3m \circ A_1t_3A_4m \circ A_4t_3A_4m))\), where \(s = M(w)\), and verify if it is greater then zero. By lemma 1 it is equivalent to verify if

\[
\frac{\delta(w, rmt_2A_1)}{\sum_{\pi_b \in \Pi: f(s, \pi_b) \neq \epsilon} \delta(w, \pi_b)}
\]

is greater then zero, where \(\Pi = rmt_2A_1 \circ rmt_2A_2 \circ rt_1A_3 \circ rt_1A_4 \circ A_1t_3A_3m \circ A_2t_3A_3m \circ A_1t_3A_4m \circ A_4t_3A_4m\).

A more sophisticated example concerns in verifying if the transmission ratings from agents \(A_1\) and \(A_2\) to the agents \(A_3\) and \(A_4\) are overheading agents \(A_3\) and \(A_4\). That is verify if the programs \(A_1t_1A_3, A_1t_1A_4, A_2t_1A_3\) and \(A_2t_1A_4\) are firing more times then \(rt_1A_3\) and \(rt_1A_4\). It is equivalent to verify if the probabilities of firing that first basic programs are greater then these last ones. So it is equivalent to verify if for a sequence \(\Lambda(A_1t_1A_3 \circ A_1t_1A_4 \circ A_2t_1A_3 \circ A_2t_1A_4)\) from an initial state \(v_1\) such that \(v_1R\Pi v_n = v_1Rv_2 \circ \cdots \circ v_{n-1}Rv_n\), where \(\Pi\) stops in state \(v_n\),

\[
\sum_{\delta(v_i, A_1t_1A_3 \circ A_2t_1A_3 \circ A_1t_1A_4 \circ A_2t_1A_4)} 1 > \sum_{\delta(v_i, rt_1A_3 \circ rt_1A_4)} 1
\]

for \(1 \leq i \leq n\) where all the involved basic problems are enabled. Determine a good firing rate for \(A_1t_1A_3, A_1t_1A_4, A_2t_1A_3\) and \(A_2t_1A_4\) is an optimisation problem for \(\Lambda(A_1t_1A_3 \circ A_1t_1A_4 \circ A_2t_1A_3 \circ A_2t_1A_4)\).

5. Conclusions and further work

In this work we presented the usage of a sound and complete logical system to multi-agent systems properties formal verification. The usage of Stochastic Petri Nets instead of the ordinary ones lets the expressive power to increase and the user will be able to model scenarios more precisely.
As it can be seen in section 4, the problem of verifying some properties is reduced to simple computations. Nevertheless the user is able to model using the graphical notion inherited from Petri Nets.

Among others, further work includes the development of a deductive system (as Natural Deduction and/or Sequent Calculus) and the investigation of the computational complexity of $DS_3$ SAT problem and its automated theorem proving.

**Acknowledgment**

The authors thanks CNPq and CAPES for the partial funding of this research.

**References**


