Framework of Designing Automata Capable of Modeling Reversibility in Concurrent and Probabilistic Environment

Arpit¹, A'fza Shafie² and Wan Fatimah Wan Ahmad¹

¹Computer and Information Sciences Department, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia
{arpittabelabux}@gmail.com, {fatimhd}@petronas.com.my
²Fundamental and Applied Sciences Department, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia. {afza}@petronas.com.my

Abstract—A framework of automata along with its detailed road-map has been proposed here. This automaton will be capable of modeling reversibility, concurrency and probabilistic environment. Along with this, motivations behind the different constructs of framework have been provided. The novelty of this framework is in respect of providing a formalism to model full non-determinism along with full probabilistic characterization irrespective of any distinctions made among states and actions, a notation of reversible transitions having a memory structure, and last but not least; a notation of bisimulation which grounds on the field of natural computing. Latter one also draws a new line of discussion.

Index Terms—Full non-determinism, Probabilistic environment and Bisimulation.

I. INTRODUCTION

Advancements in the computing environments not only arise the question of their efficient implementations but also introduce the various research dimensions. One of the dimensions which is undertaken here is modeling, verification, and analysis [9] of software systems. A very famous approach towards the modeling and verification of systems that has been significant from a long time is the modeling by finite state machine (or automata). Initially it was the only technique for modeling and analyzing the systems but the formalism like λ-calculus [1], has introduced the axiomatic approach to model the systems, and as a result, a new field known as process-algebra has been evolved. Although, various process-algebras like CCS[10], π-calculus[11], and etc., have been used today for modeling and verification purpose but they are all based on finite state-machines for their semantics; for example, CCS is based on labeled transition system (LTS) for their semantics. Even now, state-machines are not only studied for theoretical purpose but also for modeling the systems, analyzing their properties, and to verify them against certain requirements.

Various finite state-machines have been formulated till date for modeling purpose but the main concentration in this article is towards an automata, capable of modeling the systems which show stochastic characteristics during their executions, concurrent in nature, and back-track their computations; in-order-to implement the fault-tolerant strategies. Examples of such environments are various cryptographic protocols [15].

© Elsevier, 2013
(communication networks are prone to failures so such protocols also implement reversibility to ensure correct behavior), randomized distributed algorithms having fault-tolerant mechanism, and etc.

Based on various literature reviews in sec.3, it concluded that no single formalism of automata has been proposed yet, having capabilities to model concurrency, reversibility (back-tracking), and probabilistic behavior; simultaneously. Therefore, an attempt is made to provide a framework for such automata by the means of this article. The framework is proposed in the form of characteristics (using terminologies of classical set theory) that automata should possess in order to model probabilistic characteristics along with concurrency and reversibility.

II. Preliminaries

Here is a list of nomenclatures which will be used throughout the paper

1. $S$, is the set of finite states and $s, p, \ldots$, range over $S$.
2. $\Sigma$, is the set of finite actions and $\alpha, \beta, \ldots$, range over $\Sigma$.
3. $A, B, C, \ldots$, range over set of automats.
4. Let $\theta$ be any set then $\mu: \theta \rightarrow [0,1]$, $\mu$, is termed as probability distribution.

**Definition 1.** Markov chain is a pair $(S, \delta)$, where $S$ is a set of states and $\delta: S \rightarrow \mu(S)$ is a transition function $\mu(S)$ is a probability distribution over $S$.

**Bisimulation** is a theory that deals with the question, when one system is considered as a correct implementation of the other.

**Definition 2.** A relation $E$ on a set of states $S$ of a Markov chain is a bisimulation if following proposition holds $\forall (s, t) \in E \Rightarrow (\forall \mu(s \rightarrow \mu) \Rightarrow (\exists \mu'(t \rightarrow \mu \land \mu \equiv \mu')))$ (1)

Parallel composition operators serve the need of modeling synchronizations between different components of systems and provide them a single entity. Main distinctions between them are synchronous vs asynchronous.

1. **Synchronous style:** Such types of operators force its components to synchronize when they can. For example, CSP[16] provides this type of synchronization.
2. **Asynchronous style:** Such types of operators give freedom to its components either to synchronize or act independently. For example, CCS[16] has defined this type of operator.

II. Background

This section has been divided into two broad categories based on automats capable of modeling stochastic behavior and automats capable for modeling reversibility.

A. Probabilistic Automats (PAs)

Under the category of probabilistic automats (PAs), various automats have been proposed which are capable of modeling stochastic behavior and concurrency of the system but none of them are capable to implement the reversibility. Moreover, automats under this category have a lot of distinctions made among states and actions. One of the main causes of divergence among PAs is extent up-to which non-determinism is captured by PA. Non-determinism is categorized into *external* and *internal* non-determinism. In *external* non-determinism, choices are governed by the external environment; specifying several transitions with different actions leaving from the same state, whereas, *internal* non-deterministic choices are governed locally by specifying several transitions with same actions leaving from same state. Moreover, a PA is said to be a fully non-deterministic when it includes both type of non-determinism.

Although, such distinctions have their own valid reasons but they made their syntax and semantics, difficult to understand; therefore, having limited usability. Few automats, under the category of PAs are summarized below which are significant in terms of their usability and theoretical backgrounds.

**Definition 3 ([17]React).** A reactive probabilistic automata is a tuple $(S, \Sigma, \delta)$, where $\delta: S \rightarrow \mu(S) \times \Sigma$.

**Definition 4 ([17]Gen).** A generative probabilistic automata is a tuple $(S, \Sigma, \delta)$, where $\delta: S \rightarrow \mu(S \times \Sigma)$.

**Definition 5 ([3]I/O).** An I/O probabilistic automata is a tuple $(S, \Sigma, \delta)$, where $\Sigma$ is categorized into input and output actions $\Sigma \equiv \text{in} \cup \text{out}$; and $\delta$ is defines as:

\[ \delta: S \rightarrow 2^{\mu(S) \times \text{in}} \times \mu(S \times \text{out}) \]

**Definition 6 ([6]Str).** A stratified probabilistic automata is a tuple $(S, \Sigma, \delta)$, where $\delta: S \rightarrow \mu(S) \cup (S \times \Sigma)$.

**Definition 7 ([5]Vardi).** A Vardi probabilistic automata is a tuple $(S, \Sigma, \delta)$, where $\delta: S \rightarrow \mu(S) \cup 2^{S \times \Sigma}$.

**Definition 8 ([14]Seg).** A Segala probabilistic automata is a tuple $(S, \Sigma, \delta)$, where $\delta: S \rightarrow 2^{\mu(S \times \Sigma)}$.

**Definition 9 ([14]Smp-Seg).** A simple Segala probabilistic automata is a tuple $(S, \Sigma, \delta)$, where $\delta: S \rightarrow 2^{\mu(S) \times \Sigma}$.
Definition 10 ([16]PnZu). Pnueli-Zuck automata is a tuple \((S, \Sigma, \delta)\), where \(\delta: S \rightarrow 2^{2^{\sum}}\).

Definition 11 ([16]GPA). A general probabilistic automata is a tuple \((S, \Sigma, \delta)\), where \(\delta: S \rightarrow 2^{\sum}\).

Definition 12 ([4]APA). Abstract probabilistic automata is a tuple of six elements \((S, \Sigma, L, AP, V, S_0)\), where \(S_0 \in S; AP\) is a finite set of atomic propositions; \(L: S \times \Sigma \times C(S) \rightarrow 2^S\), \(C(S)\) is a set of constrain functions defined over state space; and \(V: S \rightarrow 2^S\).

On the basis of the various literatures in this domain; three different styles of parallel composition: CCS, CSP, and ACP, have been identified [16, 4]. Generally, all except few automats mentioned above use one or more styles of composition to implement parallel composition operator. The various composition styles adopted by different probabilistic automata are summarized in table.1.

<table>
<thead>
<tr>
<th>Abbreviations</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>React</td>
<td>CSP</td>
<td>-</td>
</tr>
<tr>
<td>Gen</td>
<td>ACP</td>
<td>-</td>
</tr>
<tr>
<td>I/O</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Str</td>
<td>CCS, CSP, ACP</td>
<td>-</td>
</tr>
<tr>
<td>Vardi</td>
<td>ACP</td>
<td>-</td>
</tr>
<tr>
<td>Seg</td>
<td>CCS, CSP, ACP</td>
<td>CCS, CSP, ACP</td>
</tr>
<tr>
<td>Smp-Seg</td>
<td>ACP</td>
<td>-</td>
</tr>
<tr>
<td>PnZU</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GPA</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APA</td>
<td>ACP</td>
<td>ACP</td>
</tr>
</tbody>
</table>

All probabilistic automats have used eq.1 with some minor variations for defining bisimulation relation. The only exception in this sequence is APA. APA does not describe as such any bisimulation relation but provides the comparison of its expressiveness with respect to Markov Chain (MC), and Constraint Markov Chain (CMC)[2] and it is also established that APA is as strong as MCs and CMCs in its expressive power.

B. Probabilistic Reversible Automats

Reversibility is the requirement of all fault-tolerant strategies and fault-tolerance is considered as an important characteristic of software, today. Although, many automats are there but none of them are capable of modeling reversibility along with concurrency and stochastic behavior. Here, some automats related to the field of quantum computing have been introduced which consider stochastic behavior as well as reversibility. The class in which they belong is generally abbreviated as Probabilistic Reversible Automata (PRAs). PRA is not capable of modeling concurrency, even though it has been introduced here because other automats even don’t talk about reversibility in probabilistic environment. One, process algebra named as Reversible Communicating Concurrent Systems (RCCS)[13] has introduced reversibility along-with concurrency but again environment was non-probabilistic. Now, here is a list of some PRAs.

1. PRA or doubly stochastic automata [7]
2. Probabilistic reversible decide and halt automata (DH-PRA) [8]

DH-PRA is most general, being entirely classical; the model however has a closelinks with Quantum finite automata (QFA). Moreover with some restrictions, DH-PRA may be considered as the special case of Nayaks quantum automata[12].

IV. PROPOSED SOLUTION AND DESIGN METHODOLOGY

Now, an analysis is made on various automats studied in previous section to find out their similarities and drawbacks. One point is clear from sec.3 that no PAs are able to model reversibility and no PRAs can model concurrency. Moreover, reversibility in PRA is suitable for exploring the characteristics of QFA rather than molding the reversibility induced by fault-tolerant systems. Keeping this point in mind, only the analysis of PAs has been presented in table.2

A. Proposed Framework

On the ground of the research issues identified in the beginning of sec.4, a frame-work of automata is proposed here; it will be capable of modeling reversibility in probabilistic and concurrent environment. As a first step towards the construction of this framework, a road-map is provided in fig.1 that summarizes the
issues, identified in sec.4, and provides the guidelines for the construction of proposed framework. The details of framework are given below:

<table>
<thead>
<tr>
<th>Table II. Analysis of Various Automats.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>React</td>
</tr>
<tr>
<td>Gen</td>
</tr>
<tr>
<td>I/O</td>
</tr>
<tr>
<td>Str</td>
</tr>
<tr>
<td>Vardi</td>
</tr>
<tr>
<td>Seg</td>
</tr>
<tr>
<td>Smp-Seg</td>
</tr>
<tr>
<td>PnZu</td>
</tr>
<tr>
<td>GPA</td>
</tr>
</tbody>
</table>

Fig. 1. A road-map for proposed framework

1. The mandatory components of the automats are:
   a. A set $S$ of finite states,
   b. A set $\Sigma$ of finite symbols called as input symbols or actions.
   c. A transition function $\delta : S \times \Sigma \rightarrow 2^S$

2. Probabilistic constructs: We define a function $\delta$ that takes a state $s \in S$ and a symbol $\alpha \in \Sigma$ as augs and returns a set $[\mathbb{P}_\Omega; \Omega \in \delta(s, \alpha)]$, where $\mathbb{P}_\Omega : \varepsilon_\Omega \rightarrow [0, \infty]$ is a measure function and $\varepsilon_\Omega$ is the set of all events on $\Omega$. Then, we define $\mathbb{P}_\Omega = \frac{\varepsilon_\Omega}{\varepsilon_\Omega}$ for any $\varepsilon \in \varepsilon_\Omega$. The function $\mathbb{P}_\Omega$ is called as probability density function over $\varepsilon_\Omega$.
   a. Internal Non-determinism: By the definition of $\delta$, it is clear that it allows several transitions with same input symbol leaving from same state because output of $\delta$ is a set not a single element.
   b. External Non-determinism: Augments of $\delta$ allows external non-determinism because it is defined over $S \times \Sigma$.
c. Full Probabilistic: Full probabilistic analysis can be performed by the use of the scheduler. Scheduler will probabilistically resolve the choices over next transition made by automata. Formally, we define it by 
\[ Sch: \{ \mu \in \delta(s, \alpha) | s \in S \land \alpha \in \Sigma \} \rightarrow \{0, 1\} \]

3. Reversibility constructs:
   a. \( m = < m_i | i \in I > \), is the memory in form of stack associated with automata for some indexing set \( I \); where, \( m_i = < m_k | 0 < k \leq 5 > \).
   b. \( m_1 \); will keep track of past synchronizations, \( m_2 \); will store the probability of past transition; \( m_3 \); will memorize the last taken action, and \( m_4 \) and \( m_5 \) will specify the probabilistic and nondeterministic choices available during last transition; respectively.

4. Concurrency constructs:
   a. For, parallel composition operator, \( CCS \) style has been favored because it supports asynchronous style of synchronization, and asynchronous favors non-determinism.
   b. Parallel composition operator should be associative and as well as communicative that is \( A \parallel (B \parallel C) = (A \parallel B) \parallel C \land A \parallel B = B \parallel A \), respectively.

5. Notation of \( bisimulation \) and probabilistic analysis are interleaved because \( bisimulation \) depends upon how probability space is assigned to automata. Here, an hypothesis is made to assign a probability space \( r:S \times S \rightarrow [0,1] \) such that if \( (s, p) \in S \times S \) then states must be reachable from \( s \). Moreover, if the reachability of state \( s_1 \) and state \( s_2 \) is same with respect to all other states belong to \( S \) then two states must be considered equivalent or similar. This notion of bi-similarity is distinct from eq.1 and motivation behind such \( bisimilarity \) is how efficiently a computation can reach to particular state in comparison to other states. Here, we claim that the notation of equivalence also changes with level of abstraction. This notation defines the equivalence on the uppermost level of abstraction (here, terms like level of abstraction, and abstraction; concerns with the system which is composed of sub-systems).

V. CONCLUSION

In this paper, a systematic approach to the construction of automata, capable of modeling reversibility in a probabilistic and concurrent environment, has been proposed. It comprises a road-map and a framework along with various motivations; this would be helpful in providing guidance to construct such an automata. The proposed notation of \( bisimulation \) can also be a point for further discussions.

REFERENCES


