Distinguishing Automata Machine By Using UPPAAL As A Model Checker

Yogeswaran Nagarathinam, Dr. Nor Fazlida Mohd Sani

Abstract: Upon to the evolution of technologies, electronic commerce and other online businesses are exposed to vulnerability hence invoking damages and untraceable fraud to the end users. Software engineers in the moment by moment, tracks the design and the analysis so that they can ensure the safety of the overall process from the root itself. Besides that we have proposed model checking to check on the behavior of a design. Thus our research has identified and differentiate the best of two methods of model checking which is Finite State Automata and Non Deterministic Pushdown automata. For the purpose of simulation, UPPAAL tool has been used over a part of Online Shopping system case study.

Index Terms: model checking, electronic payment, Finite State Automata, Non Deterministic Pushdown Automata, UPPAAL, Online Shopping system, possible traces

1 INTRODUCTION

Finite State Automata known by its achievement on simulating the design of a software system. Finite State Automaton has set of states which rely or response on external “inputs”. It is clearly reads sequential from start state to the end state. Finite State Automata can be categorized into two which is ‘Deterministic and Nondeterministic. Deterministic ensure the every state is in control example Deterministic automata travels only one at a time where else for non deterministic automata can be more than one state per time. These are some characteristics of Nondeterministic Finite Automata, a set of state with one start state and few final or end states. An epsilon $\Sigma$ for all possible inputs. Allowing more than one possibility states in Nondeterministic. It is said that Finite State Automata act as the most powerful way to implement logic on the applications. The fundamental is proven in many ways example checking the behavior in software design, on lexical analyzer, scanning web pages and verifying system such as transaction, stock market and for the distribution system. However they are some limitation in imposing Finite State Automata such as it can be difficult to manage in a context of a larger system without well designed example “spaghetti factor”. Secondly this process has consume lots of time since some of the design needs to rebuild again and again from scratch. Thirdly Finite State Automata has trouble in dealing with concurrency when running on multiple machine state in parallel. Apart from that, it is discovered that Finite State Automata is weak even to ensure that input string was a form of X$\in\Sigma^*$, a comparative study on Pushdown Automata has been done related to infinite words. On checking the emptiness likely Pushdown Empty Stack [1] Buchi has developed a pushdown automaton which accept infinite words. Prior to this, pushdown automata used to validate XML documents which are helpful in electronic commerce. In other word applying pushdown automata is to check infinite size of stack and to understand some non regular languages. Even though Deterministic Finite Automata can implement regular expression, Pushdown Automata likely to implement context free grammar. Pushdown Automata can store any information in stack and process it continuously with the information on top of the stack. However nondeterministic pushdown automata able to digest deterministic context free languages. It is able to move on again to the same state with similar inputs. The remainder of the paper is presribed as follows : - Section 2 would be related work, Case Study will be in Section 3 followed by Simulation in Section 4 and finally Section 5 Conclusion.

2 Related Work

Since Business Process Execution Language for Web Service always with complexity therefore semantically unclear, the researcher proposed Timed Automata [2]. A part of checking the design of particular issue pertaining to electronic commerce, model checking such as Non Deterministic Finite State used to ensure the correctness of web services. It helps to check on the design which any of transaction correlate with a time constraint. As pointed out in [2] Hybrid Timed Automata with the combination UPPAAL model used to check the Secure Electronic Transaction in regards of its capacity to check the correctness at a real time. Time based Automata also used to check in real time and probabilistic behavior in securing electronic transaction [2]. This both applies on Deterministic Finite Automata and also Non Deterministic Finite Automata. Several studies have been carried out to check the correctness of the system and the design by using model checkers. Previously researchers conduct the study of correctness by using UML statechart diagram [3]. The UML Statechart diagram is used because it can determine all possible paths of an object in the entire operation. Deepak [3] has proposed a methodology transforming UML to Finite Automata emphasizing regular grammar. To predict the valid inputs in state diagram regular grammar is highlighted. Besides that the above approach has reduced the effort, costing and also helps eliminate anomalies in software development. In addressing the issue on synthesized web services complexity which highlights on equivalence problem, validation problem and the non emptiness problem [4] the
The case study allows decidability and static analysis with aggregation on web services. Besides that, researchers identify some restrictions. Researchers propose upper bounds to the complexity of online commerce design.

Furthermore, when wrapping is in the process, it tends to wrap free lifecycles and decidability for valid artifact. However, Alex Thomo et al. [10] focuses on extensible mark up language XML, whereby XML from defined sources wraps the data. Furthermore, when wrapping is in the process, it tends to wrap diversity information from other sources. Hence, researchers defined visibly pushdown automata that supports visibly pushdown languages on approaching XML rewritings. Table 1 shows the summary of review of model checking process in electronic commerce. The summary explains the deterministic finite automata and pushdown automata which has been helping out the software engineer to do research in electronic commerce design.

### 3 METHODOLOGY

#### 3.1 CASE STUDY

In this case study section is presented in following to explore the verification method. The case study would be a part of the Online Shopping system that behavior are illustrated as such:

- The client at the initial place “Homepage” and the system direct into “SelectLtem” or second option “Login”. To illustrate further the flow starts from HomePage to SelectIttem and directed to AddToCart page here the user can do many times “SelectLtem” and “AddToCart” hence it direct to “Checkout”. Once again here the user can do the transaction many times starting from “SelectLtem” to “AddToCart” and then “Checkout”. Hence finally “LogOut” then can choose to go to the initial state “HomePage”. There are also another option after the “Checkout” the user can “Login” to access his own account. However for another option the user from the state of “HomePage” to the “Login” state and then if “LoginSuccessful” user can perform the task simultaneously “SelectLtem” to “AddToCart” then to “Checkout” finally “LogOut”. An other Option if the User has perform wrong “Login” the system direct the user to the “LoginFailed”. Furthermore if “LoginFailed” three times, the system will “BlockUser”.

In the below transition diagram model of an abstract online shopping system as given in Figure 1, elaborated as a Finite State Machine and defined as a 5 tuple as follows:

![Figure 1](image-url)

$$M = (S, \sum, s, F, T)$$

$S = \{S0,S1,....S8\}$ where the states are:-

- S0 – HomePage
- S1 – SelectItem
- S2 – AddToCart
- S3 – Checkout
- S4 – Logout
- S5 – Login, S6 LoginSuccessful, S7 – LoginFailed, S8 – BlockUser

$\sum = \{e0,e1,....e15\}$ where the events are is set of event whereby the system accept,

- $\Sigma = \{\text{Homepage, SelectItem, AddToCart, Logout, Login, LoginSuccessful, Login Failed, BlockUser}\}$ where the events are $\Sigma$

$s$ start of state system, $s$ is the initial state $s = S0 – \text{Homepage}$

$F$ set of “final” or “accepting states” $s = S4 \text{Logout}, S7 \text{LoginFailed}, S0 \text{Homepage}$

$T$ is the “transition function”, $Si+1$, and for event $ei$ where

$$T = f : S \times \sum \rightarrow S$$
<table>
<thead>
<tr>
<th>NO</th>
<th>YEAR</th>
<th>TITLE</th>
<th>PROBLEM STATEMENT</th>
<th>SOLUTION/HYPOTHESIS</th>
<th>METHODOLOGY USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2010</td>
<td>Modeling and Verifying Web Service Applications with time constraints</td>
<td>BPEL4WS with time restriction is not appropriate, to verify the processes happening in web services</td>
<td>This paper presents a formal approach to verify time-related Web Service applications defined by timed automata using Uppaal tool to simulate and verify the correctness of the system.</td>
<td>case study: airline reservation system which using time constraint. Nondeterministic finite state (Timed automata) used to ensure correctness of web services and simulate by Uppaal.</td>
</tr>
<tr>
<td>2</td>
<td>2012</td>
<td>Semantic for UML Model Transmission and Generation of Regular Grammar</td>
<td>How to verify the correctness of the design diagram and how to detect the anomalies in UML diagram</td>
<td>FSA with generation with regular grammar be used in checking the correctness in UML state diagrams.</td>
<td>Finite State Automata with regular grammar.</td>
</tr>
<tr>
<td>3</td>
<td>2008</td>
<td>Complexity and composition of synthesized web services</td>
<td>Complexity of decision problem and composition synthesis are found on web services</td>
<td>Proposed synthesized web services to uniformly manage characterize FSA and transducer abstract of web services.</td>
<td>-case study: booking travel package -FSA.</td>
</tr>
<tr>
<td>4</td>
<td>2013</td>
<td>WSST: A Tool for WS-BPEL Compositions Conformance Testing</td>
<td>In execution of web services, BPEL codes become crucial and correctness still becomes as issue</td>
<td>WSCCT tool allows online tracking test execution for correctness BPEL and Time Automata used as underlying formalism.</td>
<td>WSCCT and Timed Automata.</td>
</tr>
<tr>
<td>5</td>
<td>2009</td>
<td>Runtime Monitoring of Web Service Conversations</td>
<td>Web services are dynamic of their properties, so it’s hard to check the correctness behavior</td>
<td>Introduced a framework for monitoring runtime web services using Nondeterministic Finite Automata.</td>
<td>-Framework for runtime monitoring web services -Nondeterministics finite automata.</td>
</tr>
<tr>
<td>6</td>
<td>2008</td>
<td>A New Approach on Interactive SOA Security Model based on Automata</td>
<td>Security model built into application may not be appropriate when the application exposed as services that used by other applications.</td>
<td>ISOAS model Interactive Service Oriented Architecture Security model to highlight the secured and flexible electronic commerce.</td>
<td>-Proposed Security model called ISOAS -Finite state machine -web services.</td>
</tr>
<tr>
<td>7</td>
<td>2004</td>
<td>Automated Composition of E-services: Lookaheads</td>
<td>Complexity of constructing delegators in e services</td>
<td>General class of delegators called “lookahead” investigate complexity of constructing such delegators if they exist.</td>
<td>Wozart, automated mediator.</td>
</tr>
<tr>
<td>Year</td>
<td>2012</td>
<td>Recognizing valid artifacts in business processes</td>
<td>Recognizing violation in business rules which happen in business process</td>
<td>By considering (data)Artifacts and activities (services), identify decidability and undecidability of valid artifacts then present it on pushdown automata</td>
<td>Pushdown automata</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>--------------------------------------------------</td>
<td>---------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>2008</td>
<td>Rewriting of Visibly Pushdown Languages for XML Data Integration</td>
<td>Problem in rewriting XML data integration since data when its wrap from other source may contain diverse information</td>
<td>visibly pushdown automata that supports visibly pushdown languages on approaching XML rewritings.</td>
<td>Visibly pushdown language and visibly pushdown automata</td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>An approach to handle Real Time and Probabilistic behavior in ecommerce: validating the SET Protocol</td>
<td>Model checking on real time behavior and probabilistic way cant be handled by Uppaal tool alone.</td>
<td>Uppaal and Rapture tool allow to check the probabilistic and real-time behavior of security protocols such as Secure Electronic Transaction.</td>
<td>Time based automata Uppaal and Rapture used to verify on SET (Secure Electronic Transaction</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 1**
In the above Table 2 shows that all possibilities traces which is known as accepted inputs were found. These 16 accepting inputs compute from the model which is build shows on Figure 1. Furthermore the projection of inputs is depend from the model build by software engineers to determine the correct model which would be used in future electronic commerce. In this automaton the accepting state would be S0 Homepage, which is also initial state, S4 LogOut and S8 for Deadlock. The reading from Table 2 (possibility traces), shows that the user only have limited boundaries which can read. However the model reject the following path which is from S6(Login) to one time S7 (LoginFailed) to S8 (BlockUser). S6 (Login) to S8 (BlockUser). S1(SelectItem) to S8(BlockUser). S1(SelectItem) to S6 (Login). S3(CheckOut) to S7 (LoginFailed).

3.2 CASE STUDY OF NON DETERMINISTIC

PUSHDOWN AUTOMATA APPROACH

To distinguishing and compare the DFA automaton model, this section present case study on Non Deterministic Pushdown Automata (NDPDA). From the part of DFA illustration shown on Figure 1 the NDPA works as such :- The client have two optional Login and Select Item or Select Item and Login. From the Select item client can do multiple transaction Add To Cart and Select Item back and finally Check Out. On the other hand client Login then Select Item and Add To Cart hence Check Out. The client also can Select Item then Add To Cart then Login back. The Figure 2 below shows how the abstract NDPDA model been developed.

**FIGURE 2**

A Non Deterministic pushdown automata consist of 7 tuple, whereby:-

\[ M = (Q, \Sigma, \delta, q_0, Z_0, F) \]

\[ \delta \] \hspace{1cm} \text{is set of finite input alphabet event whereby the}
system accept, $\Sigma = \{\Sigma u(\epsilon)\}$

1 = User_Login, 2 = Select Item, 3 = Add To Cart, 4 = CheckOut

$\Gamma$ finite alphabet symbols which is \{x, y, z\}

$T$ is the "partial transition function", $Q x \Gamma = \{Q x \epsilon\} x r^*$

$T(x,y,z) = \{(S0,e0),(S1,e1),(S2,e1)\ldots\ldots,(S7,e11)\}$

S0 starting or initial states

Z0 in r (gamma) symbol on the pushdown

F contain in K is set of final states

In the below Table 3, the states in NDPDA is defined as 7

Transition which would be:

<table>
<thead>
<tr>
<th>Transition number</th>
<th>State transition</th>
<th>Input, read &amp; pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Transition 1</td>
<td>(S0 → S1)</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>From Transition 2</td>
<td>(S1 → S2)</td>
<td>$\epsilon$, $\epsilon$</td>
</tr>
<tr>
<td>From Transition 3</td>
<td>(S2 → S3)</td>
<td>$\epsilon$, $x \rightarrow \epsilon$</td>
</tr>
<tr>
<td>From Transition 4</td>
<td>(S3 → S4)</td>
<td>$\epsilon$, $x \rightarrow 2y$</td>
</tr>
<tr>
<td>From Transition 5</td>
<td>(S4 → S5)</td>
<td>$3y \rightarrow 3z$, $2y \rightarrow z$</td>
</tr>
<tr>
<td>From Transition 6</td>
<td>(S5 → S6)</td>
<td>$1z \rightarrow 1x$, $4 \epsilon \rightarrow 4$</td>
</tr>
<tr>
<td>From Transition 7</td>
<td>(S6 → S6)</td>
<td>check pop every input: $1x, y \rightarrow 1$, $2y, z \rightarrow 2$, $3y, z \rightarrow 3, 4, \epsilon \rightarrow 4$</td>
</tr>
</tbody>
</table>

In this automaton the accepting state would be S5 and S6. S0 is the initial state. The reading from Table 3 (possibility traces), shows example of five accepted inputs as such:

1) SelectItem && AddToCart && SelectItem && AddToCart && User_Login && CheckOut.

2) User_Login && SelectItem && AddToCart && SelectItem && AddToCart && User_NPDA.SelectItem && AddToCart && CheckOut.

3) SelectItem && AddToCart && User_Login && CheckOut.

4) User_Login && SelectItem && AddToCart && CheckOut.

5) SelectItem && AddToCart && User_Login && SelectItem && AddToCart && CheckOut.

After considering the languages in NDPDA and the mapping, the next section would be evaluated by UPPAAL simulation.

<table>
<thead>
<tr>
<th>Transition number</th>
<th>State transition</th>
<th>Input, read &amp; pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Transition 1</td>
<td>(S0 → S1)</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>From Transition 2</td>
<td>(S1 → S2)</td>
<td>$\epsilon$, $\epsilon$</td>
</tr>
<tr>
<td>From Transition 3</td>
<td>(S2 → S3)</td>
<td>$\epsilon$, $x \rightarrow \epsilon$</td>
</tr>
<tr>
<td>From Transition 4</td>
<td>(S3 → S4)</td>
<td>$\epsilon$, $x \rightarrow 2y$</td>
</tr>
<tr>
<td>From Transition 5</td>
<td>(S4 → S5)</td>
<td>$3y \rightarrow 3z$, $2y \rightarrow z$</td>
</tr>
<tr>
<td>From Transition 6</td>
<td>(S5 → S6)</td>
<td>$1z \rightarrow 1x$, $4 \epsilon \rightarrow 4$</td>
</tr>
<tr>
<td>From Transition 7</td>
<td>(S6 → S6)</td>
<td>check pop every input: $1x, y \rightarrow 1$, $2y, z \rightarrow 2$, $3y, z \rightarrow 3, 4, \epsilon \rightarrow 4$</td>
</tr>
</tbody>
</table>

In the following simulation model, Figure 3 represent Deterministic Finite Automata model and Figure 4 represent Non Deterministic Pushdown Automata. Furthermore in Figure 5 and Figure 6 represent verifier on both Deterministic Finite Automata and Non Deterministic Pushdown Automata. From the studies above, it can be conclude that by using model checker as UPPAAL, that can support concurrency and real time system, properties of each model can be well defined. Furthermore in this paper found that by using deterministic finite automata, it would only read certain limitation of finite inputs. The possible path or inputs could be more as we have to determine every possible steps that can be made. In the results that shown above as in verifier it can read one at per time in one particular transition however by using non-deterministic pushdown automata, the inputs are read and then put into stack before it pops out hence the inputs are read, stored and pop until the first reading which is following the concept of queue.

5 CONCLUSION

As per analyzing the proposed abstract model, it is always to be certain that the early process of design would be adequate for only allowing the valid inputs. Hence the valid inputs can be predicted by finite state automata. However Finite state automata is impossible to be used in the larger context of the design diagram and doesn’t have level of accepting
concurrency and non-deterministic process. Therefore to support the concurrency and non-deterministic process, real time based and to identify possible input, this paper suggest non-deterministic pushdown automation. As such implementing varieties of processes which includes the securable one like payment and online transaction, the model that software engineer proposed should be invulnerable to any kinds of exposure or conditions.

**FIGURE 3**

![User1 diagram](image)

**FIGURE 4**

![Push(epsilon, epsilon), Read()](image)
ACKNOWLEDGMENT
Special thanks from authors for financial support (Fundamental Research Grant Scheme, FRGS) from the Ministry of Education (MoE), Malaysia via Universiti Putra Malaysia. The principal investigator of this research project is Assoc. Prof. Dr. Nor Fazlida Mohd Sani.

REFERENCES


[5]. Afef Jmal Maalej,, Moez Krichen, Mohamed Jmaiel,: WSCCT : A Tool for WS-BPEL Compositions Conformance Testing. ReDCAD Laboratory, University of Sfax Edinburgh & Bell Labs, ACM ( March 2013 )


[10]. Alex Thom, S. Venkatesh: Rewriting of Visibly Pushdown Languages for XML Data Integration: Department of Computer Science, University of Victoria School of Computer Science, Napa Valley, California, USA, ACM 978-1-59593-991-3/08/10 (2008)