



REVUE ALGERIENNE DES SCIENCES

Section -A-

Sciences de la Nature et de la Vie

Sciences techniques

<http://univ-eltarf.dz/fr/>



An HMM_Model-Checker for complex systems

FerroumAssia^{a*}, BoudourRachid^b

^aInstitute of Computer Science, Chadli Bendjedid University, EL-Tarf, Algeria.

^b Institute of Computer Science, Badji Mokhtar University, Annaba, Algeria.

Informations	ABSTRACT
Keywords: Complex systems Probabilistic model checking HMM POCTL *Correspondance : assiaferroum@gmail.com	Nowadays, probabilistic verification for embedded systems continues to attract more and more actors in the research community. Given a formula of the logic POCTL describing the specifications of a system, an HMM model, and a mining algorithm to check if it is satisfied or not. This work presents some contributions in the verification area: the HMM as a new model, an implementation in the NetBeans 6.5 environment of a tool called "HMM_Model-Checker". Safety properties are checked on a case study of a real embedded system Hubble Space Telescope, results are encouraging.

1. Introduction

The proliferation of complex systems in all aspects of our lives places an increasing importance on the need for them to function correctly. The presence of such systems in safety-critical applications, coupled with their ever increasing complexity, means that the conventional method of checking that a system behaves as intended, testing it on a representative set of scenarios, is often inadequate.

A branch of computer science which aims to resolve this problem is formal verification. A prime example of this is probabilistic model checking [1].

Our approach differs from others in the three basic entities: model, temporal logic and probabilistic algorithm exploration.

This paper presents our tool HMM_Model-Checker followed by a case study of an embedded system which is the Hubble Space Telescope to consolidate our words.

2. Background material

2.1. Probabilistic model checking

Probabilistic model checking is an automatic formal verification technique for the analysis of

systems which exhibit stochastic behavior [2]. The technique is similar to model checking. The major difference is that a probabilistic model contains additional information on likelihood or timing of transitions between states, or to be more specific, it can model stochastic behaviour.

Probabilistic model checking refers to a range of techniques for calculating the probability of the occurrence of certain events during the execution of the system, and can be useful to establish properties such as "shutdown occurs with probability at most 0.01" and "the video frame will be delivered within 5ms with probability at least 0.97" [3].

2.2. Existing model checkers

There are a limited number of tools available for probabilistic model checking. We reference six tools of academic major. They are free and freely accessible via Internet: APMC, PRISM [2], ETMCC [4], MRMC, YMER, and VESTA.

3. Our approach

3.1 Hidden Markov Model (HMM)

An HMM [5] is a doubly embedded stochastic process with an underlying stochastic process over some state space, which is hidden. The occupied state can only be observed through another set of stochastic processes that produce a sequence of observations. Given the sequence of

observations, we do not exactly know the occupied state, but we do know the probability distribution over the set of states. This information is captured by a so-called belief state [6].

3.2 The logic POCTL

For a given HMM, one is often interested in the properties of the underlying stochastic process. In addition, one is also interested to reason about properties over the other set of stochastic processes which produce the observations. POCTL [6] allows us to specify properties of interests over HMMs.

The basic syntax for expressing a property (prop) is as follows:

(prop) ::= true | false | (expr) | (cryanceprop) |

! (prop) |
 (prop) & (prop) |
 (prop) | (prop) |
 (prop) => (prop) |

(cryanceprop) ::= $P_{(op)(p)}$ [(pathprop)] |
 ! (cryanceprop) |
 (cryanceprop) & (cryanceprop) |

(pathprop) ::= $X_{(O)}$ (prop) | (prop) U (prop) |
 (prop) U_(time) (prop) |

(time) ::= >= (t) | <= (t) |

Where:

- (expr) is an expression which evaluates to a boolean;
- (op) is a relational operator (one of <, <=, >= or >);
- (p) is an expression evaluating to a double in the range [0, 1];
- (t) is an expression evaluating to a non-negative double or integer;
- (O) is an observation.

3.3 Model checking algorithm

The algorithm to check whether an HMM satisfies a property can be adapted from the one presented by Hansson & Jonsson [7].

The algorithm is based on a recursive procedure $Sat(\Phi)$ that computes all states that satisfy the formula Φ [6].

3.4 Structure of HMM_Model-Checker

Our tool called HMM_Model-Checker is a probabilistic model checker, a tool for the modelling and analysis of systems which exhibit probabilistic behaviour.

A simplified version of the overall structure of the tool is shown in Figure 1.

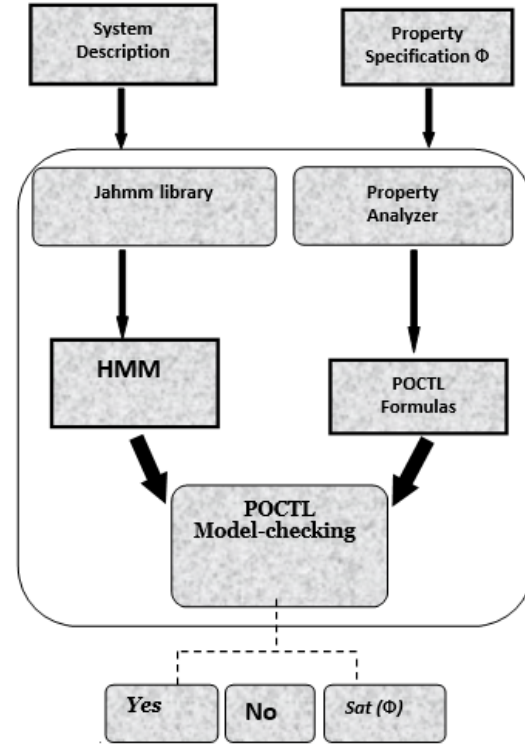


Fig. 1 :The structure of HMM_Model-Checker.

HMM_Model-Checker was implemented in the environment Netbeans and uses the Jahmm library to create the model and analyzer. Here are some key features implemented:

- Load an HMM H from a file;
- create an HMM H;
- The possibility of learning the parameters of an HMM model H using a model given and assumed to comments made by this model, we seek the transition probabilities and emission maximizing the likelihood of observations;
- Save an HMM H in a file;
- Load a sequence of observations from a file;

- Generate a sequence of observations;
- The assessment of the likelihood that the sequence of observations has been issued by a model. When multiple models exist, this evaluation allows the selection of the most likely model;
- Save a sequence of observations in a file;
- The search for this sequence of states of a model that produced the sequence of observations, i.e. the search for the hidden sequence of probabilities and that better explains these observations;
- The input screen of a formula of logic POCTL;
- Helping the user to enter the formula POCTL correctly and quickly, it displays information about the syntax of the formula and giving buttons to automatically display certain logic operators constituting the formula as: Until, Next, And so on;
- The ability to display error messages if the input formula does not meet the basic syntax of the property POCTL;
- The ability to capture the screen to comment on the property to check;
- The calculation of SAT (Φ), all states that satisfy the formula;
- Verification of $H \models \Phi$.

4. Experiment

4.1 Hubble Space Telescope (HST)

This example (an adaptation from [4]) models the failure behaviour of the Hubble space telescope. We start by describing the real system, which consists of parts that can possibly fail. Once the system is understood, we will model its failure behaviour, so that we may predict its behaviour.

The system has a steering unit with six gyroscopes, which are used to aim the telescope. Redundancy is an important issue when designing systems such as the space telescope, since it is obvious that performing repairs is not trivial. This is why the telescope is designed in such a way that it will still function with full accuracy when only three of its six gyroscopes are operational. With less than three gyroscopes the telescope turns into sleep mode, meaning repairs are necessary. If none of the gyroscopes are operational the telescope will crash. No repair mission will be undertaken, as long as there are more than two gyroscopes operational [8].

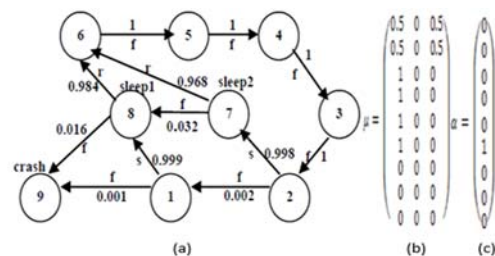
4.2 The model HMM

The model HMM of the system that has just been described is depicted in Figure 2. The model has a total of nine states. Each state has a number and label (shown respectively inside and outside the state symbol).

States 1 through 6 are not labelled explicitly; their labels equal their state number, which represents the number of operational gyroscopes. States 7, 8, and 9 are labelled sleep2, sleep1 and crash.

State 6 is considered the initial state where all gyroscopes are operational; this is also the state of the telescope after a successful repair mission. The state with label crash is a terminating state.

The model HMM has three observations: *f* represents a failure event of a gyroscope, *r* means a repair mission has been undertaken successfully and *s* means the telescope is going to sleep.



(a) The transition system with the set of observations
(b) Observation probability matrix
(c) The initial distribution

Fig. 2 :The HMM of the HST.

4.3 Some properties to be checked

4.3.1 Description

Among the properties satisfied during testing by our tool, we will mention three :

- The probability that the next observation is f and then the model don't go to state 6 meets the bound ≤ 1 ;
- The probability that the telescope will not crash meets the bound ≤ 1 ;
- The probability that the telescope will crash is equal to 0.

4.3.2 Specification In POCTL

The formulas POCTL are respectively as follows:

- $P \leq 1 [X F \neg(\text{six})]$;
- $P \leq 1 [X F \neg(\text{crash})]$;
- $P \geq 0 [X E (\text{crash})]$.

5. Results And Comparisons

We compared the results obtained by HMM_Model-Checker to double compared to the same system HST: the first with those obtained manually and the second to those obtained by the tool ETMCC [4]. The results obtained are summarized in Table 1 and are all identical.

Table-1. Comparative results.

Property	Manual approach	ETMCC	HMM_Model-Checker
1	verified	Verified	verified
2	verified	Verified	verified
3	verified	Verified	verified

6. Conclusion

In this paper, we focused on one aspect of automatic verification based on probabilistic model, a technique more general and realistic, which continues to attract more followers in the audit universe. We designed and implemented a tool called HMM_Model-Checker. It allows verifying properties defined by logical

POCTL over an HMM model with a very friendly interface. It was tested on a case study of an embedded system which is the HST and the results are encouraging.

In the future, we hope to develop new features, on the one hand extend PRISM models to our tools such as DTMC, CTMC to our tool and on the other and add other properties to experiment on new real applications.

References :

- [1] Parker, D. A., August 2002. Implementation of symbolic model-checking for probabilistic systems. A thesis submitted to the Faculty of Science of the University of Birmingham for the degree of Doctor of Philosophy. School of Computer Science, Faculty of Science, University of Birmingham.
- [2] Hinton, A., Kwiatkowska, M., Norman, G., Parker, D., 2006. PRISM: A tool for automatic verification of probabilistic systems. In Holger Hermanns and Jens Palsberg, editors, Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06), volume 3920 of LNCS, pages 441–444, Berlin. Springer.
- [3] Kwiatkowska, M. Z., Norman, G., Parker, D., 2002. PRISM: Probabilistic symbolic model checker. In Tony Field, Peter G. Harrison, Jeremy T. Bradley, and Uli Harder, editors, Proc. 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS'02), volume 2324 of LNCS, pages 200–204, Berlin. Springer.
- [4] Hermanns, H., Katoen, J.-P., Meyer-Kayser, J., Siegle, M., 2003. A tool for model-checking Markov chains. Int. J. on Softw. for Technology Transfer (STTT), 4(2):153–172.
- [5] Rabiner. L., R., February 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 77(2):257–286.
- [6] Zhang, L., Hermanns, H., Jansen, D. N., May 2005. Logic and Model Checking for Hidden Markov Chains, AVACS Technical Report No. 6, SFB/TR 14 AVACS. ISSN: 1860-9821.
- [7] Hansson, H., Jonsson, B., 1994. A logic for reasoning about time and probability. Formal Aspects of Computing, 6(5):512–535.
- [8] Oldenkamp, H., A., May, 2007. Probabilistic model checking -A comparison of tools-. Masters Thesis in Computer Science, University of Twente, Faculty EEMCS, Computer Science Department, Formal Methods and Tools Group.