Marimba: A Tool for Verifying Properties of Hidden Markov Models

Rong Li [1,2, a], Lei Quan [3,b]

[1] East China Institute of Technology,Jiangxi NanChang in China

[2]Guangxi Key Laboratory of Spatial Information and Geomatics,Guilin in China

[3] East China Institute of Technology,Jiangxi FuZhou in China

[a]rli@ecit.cn, [b]lquan@ecit.cn

Keywords:. Automated Technology, Robotics, Human-robot interaction ,Hidden Markov Models.

Abstract. The formal verification of properties of Hidden Markov Models (HMMs) is highly desirable for gaining confidence in the correctness of the model and the corresponding system. A significant step towards HMM verification was the development by Zhang et al. of a family of logics for verifying HMMs, called POCTL*, and its model checking algorithm. As far as we know, the verification tool we present here is the first one based on Zhang et al.'s approach. As an example of its e effective application, we verify properties of a handover task in the context of human-robot interaction. Our tool was implemented in Haskell, and the experimental evaluation was performed using the humanoid robot Bert2.

Introduction

A Hidden Markov Model (HMM) is an extension of a Discrete Time Markov Chain (DTMC) where the states of the model are hidden but the observations are visible. Typically, an HMM is studied with respect to the three basic problems examined by Rabiner in [9]. However, to the best of our knowledge, no practical model checker exists for HMMs despite their broad range of applications, e.g., speech recognition, DNA sequence analysis, text recognition and robot control. We describe in this paper a tool for verifying HMM properties written in the Probabilistic Observation Computational Tree Logic* (POCTL* [11]), and use this tool for verifying properties of a robot-to-human handover interaction.

POCTL* is a specification language for HMM properties. It is a probabilistic version of CTL* where a set of observations is attached to the next operator. Zhang et al. [11] sketched two model checking algorithms for POCTL*, an "automaton based" approach, and a "direct" approach. We opted for the direct approach for its lower time complexity. Noticeably, this approach produces a DTMC D and a Linear Temporal Logic (LTL) formulaφ , so the PRISM [6] model checker could be used to verify this property. Such a model checker follows the automata based approach whose complexity is doubly exponential in |D| and polynomial in |D|, whereas we implemented the direct method by Courcoubetis et al. [1] whose complexity is singly exponential in j j and polynomial in |D|, which is also the final complexity of our tool. This direct method repeatedly constructs a DTMC and rewrites an LTL formula, such that one temporal operator is removed each time while preserving the probability of satisfaction.

We have named our model checker Marimba. A marimba is a xylophone-like musical instrument that is popular in south-east Mexico and Central America. Marimba [5] was implemented in Haskell and compiled with GHCi. Our tool is available for download from https://github.com/nobernan/Marimba.

Tool architecture and implementation

Haskell was chosen to code this first version of Marimba since it allows us to work in a high-level abstract layer, by providing useful mechanisms like lazy evaluation and a pure functional paradigm. Furthermore, Haskell manages recursion efficiently; this is a valuable aspect because recursive calls are made continuously throughout the execution. As a future work, we consider coding Marimba in a language like Java and make it a symbolic model checker.

Marimba features a command-line interface. Furthermore, instead of working with a command window, a more user friendly and preferable execution is accomplished through the *Emacs* text editor extended with the Haskell-mode.

Marimba's input and modules

The first input is a .poctl file with the six elements of an HMM H, namely a finite set of states S, a state transition probability matrix A, a finite set of observations , an observation probability matrix B, a function L that maps states to sets of atomic propositions from a set AP_H, and an initial probability distribution π over S. The second input is a POCTL* state formula Φtyped in the command window according to the syntactic rules:
Φ ::=true | false | a | (-Φ) | ($\Phi \vee \Phi$) | ($\Phi \wedge \Phi$) | ($P_{\bowtie p}(\Phi)$),
Φ ::= Φ | (-Φ) | ($\Phi \vee \Phi$) | ($\Phi \wedge \Phi$) | ($X_o\Phi$) | ($\Phi u^{\leq n} \Phi$) | ($\Phi u \Phi$),
Where $a \in AP_H, o \in \theta, n \in N, p \in [0,1]$,and $\bowtie \in \{\leq, <, \geq, >\}$. In addition, we define $X\Omega\varphi$ as a shorthand for $\vee\ O \in \Omega\ X_o\varphi$ provided $\Omega \subseteq \theta$. We examine below the six Haskell modules that constitute Marimba.

ModelChecker.hs performs the initial computations of the model checker for POCTL*. It recursively finds a most nested state subformula of Φ , not being a propositional variable, and the states of H that satisfy it. On the one hand, finding the states satisfying a propositional subformula is straightforward. On the other hand, we invoke the module DirectApproach.hs to obtain the states satisfying a probabilistic state subformula. Next, this module extends the labels of such states with a new atomic proposition a. In Φ, the state subformula being addressed is replaced by a. The base case occurs when we reach a propositional variable, so we return the states that have it in their label.

DirectApproach.hs transforms the HMM H into a DTMC D, and removes from the specification the observation set attached to the next operator X by generating a conjunction of the observation-free X with a new propositional variable. Thus, we obtain an LTL formula that is passed, together with D, to the module Courcoubetis.hs. The new propositional variables are drawn from the power set of observations. Remarkably, it is not necessary to compute such a power set since the label of a state in D is easily calculated.

Courcoubetis.hs implements a modified version of the method by Courcoubetis et al. to find the probability that an LTL formula is satisfied in a DTMC. In this module, when dealing with the U and $U^{\leq n}$ operators, we apply ideas from [10] for computing a partition of states of D. Moreover, to handle the U operator we have to solve a linear equation system. To that end, we use the linearEqSolver library [3], which in turn executes the Z3 theorem prover [2].

Lexer.hs and Parser.hs are in charge of the syntactic analysis of the input. Finally, Main.hs is loaded to start Marimba. This module manages the interaction with the user, and starts the computation by passing control to ModelChecker.hs.

In a typical execution, Marimba prompts the user to enter a .poctl le path. Next, our tool asks whether or not the user wants to take into account the initial distribution in the computation of the probability of satisfaction. This choice corresponds to opposite ideas presented in [1] and [11], i.e., the method by Courcoubetis et al. uses the initial distribution to de ne their probability measure, contrary to that de ned by Zhang et al. Afterwards, a POCTL* formula has to be entered. Marimba returns the list of states satisfying this formula, and asks the user whether there are more formulas to be verified on the same model.

The .poctl le is simply a text le where the elements of an HMM are defined, e.g., the set of states is defined by the reserved word States, and if the model consists of five states, we write States=5.

Likewise, POCTL* formulas have a natural writing, for example, $P_{<0.1}(X_{\{o1\}}a)$ is typed as P[<0.1](X_{1}a).

Verification of a human-robot interaction

We applied Marimba to a real-world example, namely the verification of the robot-to-human handover task [4] using the robot Bert2 [7] at the Bristol Robotics Laboratory (BRL). The robot's decision to release the object during the handover task is determined by an HMM [4]. Figure 1 presents the state diagram of the HMM corresponding to the basic handover interaction, where the label L(s) is defined for each state.

Next, we initialise A, B and π of the HMM as follows. The process starts at state Robot not hold, so its initial distribution value π_1 is almost one, while the other states have initial distribution values close to zero. The initial matrix A must encourage the transitions shown in Figure 1. To initialise B, we consider as observations the ordered pairs whose first and second components are the index and middle finger metacarpophalangeal joint motor current values, respectively. By the Cartesian product of these values, we obtain 56,404 observations. Since these observations are merged with the states to generate the DTMC passed to Courcoubetis.hs, and the size of a formula could grow considerably by associating the next operator with up to 56,404 observations, Marimba's execution is not practical under these circumstances. Vector quantisation 8 was used to reduce the number of observations to just 13, which were taken to initialise matrix B. Thus, the initial ordered pairs are grouped into 13 regions of the plane representing the observations.

To make reliable estimates, we collected observations from 50 handover ex-periments on Bert2. These observations were used to train the initial HMM with the reestimation method found in the solution of Rabiner's Problem 3 9.

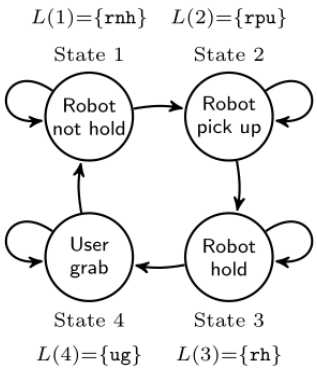

Fig. 1 The labelled states involved in the basic handover process.

```
Main> main
Enter the file name where the HMM is located.
examples/ModelBert2.poctl
Would you like to consider each state as if it were the initial
state, i.e., as if it had initial distribution value equal to 1? y/n: y
Enter the POCTL* formula we are interested in.
P[>0.88] (X_{3,4,6}(X_{3,4,6}(X_{3,4,11}(X_{3,4,11}T))))
The states that satisfy it are:
(Probability of satisfaction of each state:[4.998198505964186e-10,
            4.08659792160621e-6,7.508994137303159e-3,0.8915357419467848])
[4]
Do you want to continue checking more specifications? y/n: n
```

Fig. 2 Verifying a property with Marimba.

Liveness properties. A liveness property requires that a good thing happens during the execution of a system. For example, we would like to know whether the model generates the sequence of observations O = o1, o2, o3, o4 where o1, o2 $\in$ {3, 4,6} and o3, o4 $\in$ {3, 4, 11}, with probability greater than 0.88, that is,$P>0.88(X_{\{3,\ 4,6\ \}}(X_{\{3,\ 4,6\ \}}\ (X_{\{3,\ 4,\ 11\}}\ (X_{\{3,\ 4,\ 11\}}\text{true}))))$. Interestingly, this property is a generalisation of Rabiner's Problem 1 9. Marimba's execution for this property is found in Figure 2. The inputs are the trained HMM, de ned in ModelBert2.poctl, and the previous formula. The output returned by Marimba is State 4. Hence, the model starting at state User grab is likely to generate O.

A second liveness property states that with probability at least 0.9, Bert2 releases the object when the user grabs it. The POCTL* formula for this property is $P\geq_{0.9}(\text{rh} \wedge (\text{rh u (ug} \wedge \text{ug u rnh})))$. Marimba

outputs State 3, i.e., the specication is satis ed when the starting state is Robot hold. So, we expect Bert2 to hold the object, and let it go when the user grabs it.

Conclusions

Since the automatic veri cation of properties of HMMs seems to be an unat-tended problem, we present here Marimba, a Haskell implementation of the model checking algorithm for POCTL* [11]. This model checking algorithm was slightly modi ed to carry out its computations in a real program. Marimba's calculation is basically broken out in three stages that are coded in the modules

ModelChecker.hs, DirectApproach.hs and Courcoubetis.hs, such that the involved components, steps and transformations are well arranged throughout the im-plementation. Finally, we have successfully applied Marimba to verify relevant properties of a handover interaction from the robot Bert2 to a human.

References

[1] C. Courcoubetis and M. Yannakakis, The complexity of probabilistic veri cation, J. ACM 42 (1995), no. 4, 857-907.

[2] L. De Moura and N. Bj rner, Z3: An e cient SMT solver, Proceedings of the Theory and Practice of Software (TACAS '08), LNCS, Springer, 2008, pp. 337-340.

[3] L. Erkok, linearEqSolver: a library to solve systems of linear equations, using SMT solvers., https://github.com/LeventErkok/linearEqSolver.

[4] E. C. Grigore, K. Eder, A. G. Pipe, C. Melhuish, and U. Leonards, Joint action understanding improves robot-to-human object handover, IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 4622-4629.

[5] N. Hernandez, Model checking based on the hidden Markov model and its application to human-robot interaction, Master's thesis, Universidad Nacional Autonoma de Mexico, Mexico, 2014, Available from http://132.248.9.195/ ptd2014/noviembre/303087692/Index.html.

[6] M. Kwiatkowska, G. Norman, and D. Parker, PRISM 4.0: Veri cation of proba-bilistic real-time systems, Proc. 23rd International Conference on Computer Aided Veri cation (CAV '11), LNCS, vol. 6806, Springer, 2011, pp. 585-591.

[7] A. Lenz, S. Skachek, K. Hamann, J. Steinwender, A. G. Pipe, and C. Melhuish, The BERT2 infrastructure: An integrated system for the study of human-robot in-teraction, 10th IEEE-RAS International Conference on Humanoid Robots, IEEE, 2010, pp. 346-351.

[8] Y. Linde, A. Buzo, and R. M. Gray, An algorithm for vector quantizer design, IEEE Transactions on Communications 28 (1980), 84-95.

[9] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (1989), 257-286.

[10] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker, Mathematical techniques for analyzing concurrent and probabilistic systems, CRM Monograph Series, vol. 23, American Mathematical Society, 2004.

Supported by Key Laboratory of Spatial Information and Geomatics(Guilin University of Technology)(1103108-26)