Marimba
User's Manual

Noe Hernandez
no.hernan@gmail.com, no_hernan@ciencias.unam.mx

April 29, 2015

Marimba is a model checker for hidden Markov models (HMMs) whose specification language is the logic POCTL* [4, 5]. Marimba is also the name of a musical instrument popular in southeast Mexico and Central America, it is similar to a xylophone.

Our model checker, i.e., Marimba, is coded in Haskell and consists of six modules:

- *Courcoubetis.hs*
- *DirectApproach.hs*
- *Lexer.hs*
- *Main.hs*
- *ModelChecker.hs*
- *Parser.hs*

1 How to use Marimba

Marimba starts by loading the *Main.hs* module that contains the function `main`, which is called to start the model checker. Next, some GNU licensing text is shown, as well as the sentence: `Enter the file name where the HMM is located.` (see Figure 1). The program now waits for a *.poctl* file to be passed to it.

```
Prelude> :l Main.hs

*Main> main

Marimba  Copyright (C) 2014  Noe Hernandez
This program comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
For details see the COPYING file enclosed with the source code.

Enter the file name where the HMM is located.
```

Figure 1: Marimba loading and initialisation processes.

The six elements of an HMM have to be defined within a *.poctl* file. We use the reserved words: `States`, `Transitions`, `Observations`, `ObsProb`, `Labelling` and `Initial`, to define the set of states, the transition probability matrix, the set of observations, the observation probability matrix, the labelling function and the initial distribution, respectively. Moreover, in the *.poctl* file the sets of states and observations are given as integer numbers representing their size, the probability matrices for the state transitions and the observation distributions are expressed as lists of lists of double values, the labelling function is a list of lists of strings (letters only), and the initial distribution is a list of double values. We require

the definition of the number of states (identified by the reserved word `States`) to be the first element defined within the *.poctl* file. Likewise, the number of observations (an integer identified by the reserved word `Observations`) has to be defined before the observation probability matrix. Apart from these two constraints, the rest of the HMM elements can be defined regardless of their position within the *.poctl* file. For example, consider the *ModelChef.poctl* file found in the *examples* folder, and depicted in Figure 2. This model is defined in detail in Chapter 1 of [1], and describes a chef's mood variations with time where the main courses he or she cooks are considered as observations of the HMM.

```
States = 2

Transitions = [[0.7, 0.3], [0.4, 0.6]]

Labelling = [["c"],["d"]]

Observations = 3

ObsProb = [[0.5, 0.3, 0.2],
           [0.1, 0.6, 0.3]]

Initial = [0.6, 0.4]
```

Figure 2: The *ModelChef.poctl* file.

Observe that in this case the set of states is $\{1, 2\}$. In general, if `States = n`, the set of states is $\{1, 2, ..., n\}$. Likewise, if `Observations = m`, the set of observations is $\{1, 2, ..., m\}$. When asked for a file where an HMM is located, we might answer

```
Enter the file name where the HMM is located.
examples/ModelChef.poctl
```

Then, Marimba asks `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:`. If the answer is `y`, Marimba regards each state as if it were the one the system starts from, and computes the probability of satisfaction accordingly. If the answer is `n`, Marimba computes the probability of satisfaction of each state by taking into account the likelihood that such a state is initially chosen as specified by the initial distribution. Let us say that we answer this question with `y`.

Next, the model checker requests a POCTL* state formula by printing

```
Enter the POCTL* formula we are interested in.
```

To type POCTL* formulas, we associate to each grammar's terminal symbol a natural encoding as shown below:

Terminal Symbol	**New encoding**
Boolean value true	Reserved letter `T`
Boolean value false	Reserved letter `F`
Atomic proposition a	Sequence of non-reserved letters
Negation operator $\neg$	Symbol `~`
Disjunction operator $\vee$	Reserved letter `v`
Conjunction operator $\wedge$	Symbol `^`
Probabilistic operator $\mathcal{P}_{\bowtie p}$	`P[`$\bowtie$ p`]`, where `P` is a reserved letter
Next operator $\mathbf{X}_{\{m_1,\ldots,m_i\}}$	`X_`$\{m_1, \ldots, m_i\}$, where `X` is a reserved letter
Bounded until operator $\mathcal{U}^{\leq n}$	`U` n, where `U` is a reserved letter
Unbounded until operator $\mathcal{U}$	Reserved letter `U`
Left Parenthesis (	Symbol `(`
Right Parenthesis)	Symbol `)`

Observe that,

- The encoding of the comparison symbol $\bowtie$ can take any of the values `<`, `<=`, `>=` or `>`.
- For the threshold *double* value p used by the probabilistic operator, the inequality $0 \leq p \leq 1$ holds.
- Since we are considering the observations as integer numbers from 1 to the value of `Observations` defined in *ModelChef.poctl*, every element in the subset of observations $\{m_1, \ldots, m_i\}$, attached to the next operator, also lies within this interval.
- The bound n used in $\mathcal{U}^{\leq n}$ is a non-negative integer.

Thus, when asked for a formula to verify, we might type:

```
Enter the POCTL* formula we are interested in.
P[>= 0.3]((d ^ ~X_{1}T) U 10 (c ^ X_{1}T))
```

which indicates that with probability at least 0.3 the chef will be upset up to the day he/she cooks steak, such a day could be today or any of the next ten days (see the example located in Subsection 2.4 of this document for further details).

The model checking algorithm is executed on the already passed HMM and this POCTL* formula. If the specification involves the probabilistic operator, Marimba prints an intermediate result in parentheses consisting of a list with the probabilities of each state satisfying the argument of the probabilistic operator, as it can be seen in Figure 3. Finally, Marimba returns the list of states satisfying the input formula. The output for the particular HMM and POCTL* formula given above is:

```
The states that satisfy it are:
(Probability of satisfaction of each state: [0.5,0.39047935101157205])
[1,2]
```

Figure 3: Output returned by Marimba.

Therefore, states 1 and 2 of the HMM defined in *ModelChef.poctl* satisfy the formula $\mathcal{P}_{\geq 0.3}((d \wedge \neg \mathbf{X}_{\{1\}}\mathsf{true})\ \mathcal{U}^{\leq 10}(c \wedge \mathbf{X}_{\{1\}}\mathsf{true}))$.

Afterwards, Marimba asks whether the user wants to keep verifying properties on this same model. It prints: `Do you want to continue checking more specifications? y/n:`. If the answer is `y`, the program waits for a new formula to be entered. If we respond with an `n`, the tool finishes.

2 Examples

In the *examples* folder enclosed with the Marimba's source code we find several instances of hidden Markov models and POCTL* specifications that were verified with our tool.

2.1 Bert2

The file *examples/ModelBert2.poctl* contains the HMM corresponding to the basic handover interaction from Bert2 to a human described in Section 3 of our paper proposal [2]. In the file *examples/Modelbert2.txt*, we have written down the specifications studied in such a proposal. Marimba's execution for this model and specifications is shown in Figure 4.

```
*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: y

Enter the POCTL* formula we are interested in.
P[>=0.9] (rh ^ (rh U (ug ^ ug U rnh)))

The states that satisfy it are:
(Probability of satisfaction of each state: [0.0,0.0,1.0,0.0])
[3]

Do you want to continue checking more specifications? y/n: y

Enter the POCTL* formula we are interested in.
P[<0.05] (rh ^ X_{1,2,3,4,5,6,7,8,9,10,11,12,13}(rnh v rpu))

The states that satisfy it are:
(Probability of satisfaction of each state: [0.0,0.0,0.0,0.0])
[1,2,3,4]

Do you want to continue checking more specifications? y/n: n
*Main>
```

Figure 4: Using Marimba for verifying a basic handover interaction on Bert2.

2.2 PCTL verification on a DTMC

For a second usage example of Marimba we take the model and specifications found in Chapter 2 of [3], where probabilistic verification is studied on a Discrete Time Markov Chain (DTMC) and PCTL is used as the specification language. In such a chapter, a concrete DTMC is given as part of an example, so we defined in *examples/ModelKw.poctl* this DTMC as an HMM with one trivial observation. Moreover, on this DTMC three properties are examined, which we have written as POCTL* formulas in the file *examples/ModelKw.txt*. Taking into account that Marimba's initial state is 1 and in [3] the initial state is 0, the results obtained with our tool (see Figure 5) were the same as the ones computed in [3].

```
*Main> main

Enter the file name where the HMM is located.
examples/ModelKw.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[>=0.9](X_{1}(~try v succ))

The states that satisfy it are:
(Probability of satisfaction of each state: [0.0,0.99,1.0,1.0])
[2,3,4]

Do you want to continue checking more specifications? y/n: y

Enter the POCTL* formula we are interested in.
P[>0.98] (T U 2 succ)

The states that satisfy it are:
(Probability of satisfaction of each state: [0.98,0.9898,0.0,1.0])
[2,4]

Do you want to continue checking more specifications? y/n: y

Enter the POCTL* formula we are interested in.
P[>0.99](try U succ)

The states that satisfy it are:
(Probability of satisfaction of each state: [0.0,0.98989898989899,0.0,1.0])
[4]

Do you want to continue checking more specifications? y/n: n
*Main>
```

Figure 5: Verifying PCTL properties on a DTMC with Marimba.

2.3 Zhang's example

In Chapter 5 of [4], Zhang gives the details of a model checking algorithm for POCTL*, and provides an example of such verification process. In Figure 6, we have tested Marimba with this example too, obtaining the same result. Thus, the HMM and specification corresponding to Zhang's example are defined in *examples/ModelZhang.poctl* and *examples/ModelZhang.txt*, respectively. Observe that in [4] states 0 and 4 are returned as the states satisfying the specification, whereas Marimba outputs states 1 and 5. The reason of this mismatch is that in Marimba the state index starts at 1, and Zhang begins counting states at 0.

```
*Main> main

Enter the file name where the HMM is located.
examples/ModelZhang.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.
(~b) ^ P[<0.05] (a U (X_{1}b))

The states that satisfy it are:
(Probability of satisfaction of each state: [4.1666666666665e-2,8.333333333333083e-2,
                                             0.16666666666666333,0.16666666666666333,0.0])
[1,5]

Do you want to continue checking more specifications? y/n: n
*Main>
```

Figure 6: Zhang's example run in Marimba.

2.4 Chef's mood problem

Our last example is found in Chapter 1 of [1] where a chef's mood variations with time are modelled by the HMM defined in *examples/ModelChef.poctl*, such that the main courses the chef prepares are considered as observations. Here, we verified three properties. First, we want to determine whether the chef will be *upset* up to the day he/she cooks *steak*, which might occur today or in any of the next ten days, with probability at least 0.3, i.e., $\mathcal{P}_{\geq 0.3}((d \wedge \neg \mathbf{X}_{\{1\}}\mathsf{true})\ \mathcal{U}^{\leq 10}(c \wedge \mathbf{X}_{\{1\}}\mathsf{true}))$, where the atomic propositions c and d represent the *happy* and *upset* states, respectively, and the observations 1, 2 and 3 are associated with the meals *steak*, *pasta* and *haddock fillet*, respectively. Second, with probability less than 0.1, the chef prepares *pasta* today being *upset*, and tomorrow prepares *haddock fillet* being still *upset*, i.e., $\mathcal{P}_{<0.1}(d \wedge \mathbf{X}_{\{2\}}(d \wedge \mathbf{X}_{\{3\}}\mathsf{true}))$. Finally, the third property is taken from Chapter 3 of [1], it is $\neg c \wedge \mathcal{P}_{>0.5}(\mathbf{X}_{\{1,3\}}d)$. Marimba's execution for these model and properties is given in Figure 7. Note that for this execution, the initial distribution values are considered when computing the probability of satisfaction of each state. Hence, we answered `n` to the question `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:`.

```
*Main> main

Enter the file name where the HMM is located.
examples/ModelChef.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: n

Enter the POCTL* formula we are interested in.
P[>= 0.3]((d ^ ~X_{1}T) U 10 (c ^ X_{1}T))

The states that satisfy it are:
(Probability of satisfaction of each state: [0.3,0.15619174040462883])
[1]

Do you want to continue checking more specifications? y/n: y

Enter the POCTL* formula we are interested in.
P[<0.1](d ^ X_{2}(d ^ X_{3}(T)))

The states that satisfy it are:
(Probability of satisfaction of each state: [0.0,4.3199999999999995e-2])
[1,2]

Do you want to continue checking more specifications? y/n: y

Enter the POCTL* formula we are interested in.
~c ^ P[>0.05](X_{1,3}d)

The states that satisfy it are:
(Probability of satisfaction of each state: [0.126,9.6e-2])
[2]

Do you want to continue checking more specifications? y/n: n
*Main>
```

Figure 7: Verification of the chef's mood problem with Marimba.

3 Authors

Noé Hernández	no.hernan@gmail.com no_hernan@ciencias.unam.mx nohernan@turing.iimas.unam.mx

4 Installation

An installation document comes together with the source code of Marimba.

5 Copying / License

Marimba is free software distributed under the GNU General Public License. A copy of such a license is included in Marimba's source code in the COPYING file.

References

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

[2] N. Hernandez, K. Eder, E. Magid, J. Savage, and D. Rosenblueth. Marimba: A tool for verifying properties of hidden Markov models, 2015.

[3] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems,* P. Panangaden and F. van Breugel (eds.), volume 23 of *CRM Monograph Series*. American Mathematical Society, 2004.

[4] L. Zhang. Logic and model checking for hidden Markov models. Master's thesis, Saarland University, Germany, 2004.

[5] L. Zhang, H. Hermanns, and D. N. Jansen. Logic and model checking for hidden Markov models. In *Formal Techniques for Networked and Distributed Systems - 25th International Conference*, volume 3731 of *Lecture Notes in Computer Science*, pages 98–112. Springer, 2005.