Contents

Introduction

EpTempMC is a tool designed to perform model checking of the temporal extensions of the Exogenous Probabilistic Propositional Logic (EPPL): Exogenous Probabilistic Computational Tree Logic (EpCTL) and Exogenous Probabilistic Linear Time Logic (EpLTL). The objective of this document is to be used as a manual for the tool. As such, extensive descriptions of the logics or their properties will not be presented. If you are unfamiliar with the base logic EPPL, with the concept of temporal logic or with model checking procedures, you should probably read some of the documentation first.

EpTempMC is still in development; a beta version, if you will. There are identified issues that will be corrected and functionalities that will be added, but it was deemed reasonably useful to be released. It is composed of three independent parts, one of which is a third party application:

- A Graphical User Interface: As the name implies, the GUI performs the interaction between the user and the other modules in an intuitive and aesthetically appealing way. It allows for an intuitive description of formulae and probabilistic Kripke Structures. Although all results are presented on screen, they are also saved in independent files so that they can be re-inputed to the other modules if needed. If you need to check a particularly complex formula or model, you can skip this part and script your own input file. This part was programmed in C++ and the main concerns were to provide an intuitive, easy to use and aesthetically pleasing interface. Most updates on this module can be expected to be improvements on these aspects.

- A Kernel: This is where all EPPLcomputations are preformed and reduction to classical model checking is achieved. It bridges the GUI and the NuSMV, taking global EpCTL and EpLTL formulae and Probabilistic Kripke Structures, converting them to classical NuSMV models and interpreting the output back to EpTemp context. This part was programmed in C and efficiency was the main concern. It uses the CUDD package, an excellent BDD manipulation package developed at Colorado University. Most updates on this module can be expected to be improvements on efficiency (other than bug correcting, of course).

- NuSMV: NuSMV is a symbolic model checking tool for classical temporal logic, an extension of the CMU SMV. We use NuSMV as a black box to perform all the temporal calculi. The tool and its documentation are available here.

Installation, Setup and Compatibility

Currently, EpTempMC is only available for Windows OS. While originally developed for Unix systems, early in development it was decided that Bayesian Networks (an essential part of the states description) would be represented in MSBNX, an application that runs only in Windows OS. Therefore, all the components were developed with Windows compatibility in mind. There are plans to, at some point in time, adapt the program to run in Unix OS as well, with a different BN representation software.

The EpTempMC can be found herePLACEHOLDER, it comes as an usual Windows installation package. In addition to EpTempMC , you will need NuSMV, a very good third party model checking tool that can be found here and MSBNX, a Microsoft application that allows for easy representation of Bayesian Networks that can be found here.

After downloading and installing NuSMV, you must either install EpTempMC to the same directory or add the NuSMV \version_number \bin directory to the PATH. This is an important step, as EpTempMC needs to call NuSMV and the program won't work at all if you do not do this. If you do not know how to change the PATH, click here.

In some cases, non-ASCII symbols from the interface are not presented correctly. This is probably because the version of the font (in this case ARIAL) in your computer is not updated. Figure 1 is a snapshot of how the interface looks. If you have this problem you can simply update the font or send me an email (`djhenriques@gmail.com`) and I'll gladly send you a version of the interface only with ASCII symbols.

Very brief description of the logics

EPPL is a very expressive logic that allows for probabilistic quantitative reasoning at state level. The logic and its temporal extensions (hereinafter called collectively EpTemp) are thoroughly discussed in PLACEHOLDER and as such we will only describe them very briefly here.

The syntax of EpTemp , consists on formulae at two levels: we use propositional language as a base language and then use its elements as terms to define another language, at an higher level.

As such, we consider *basic formulae*, propositional formulas over a finite set of propositional symbols Λ, *terms*, a set that represents (algebraic) real numbers,

allowing for quantitative reasoning and *global formulae*, which are essentially propositional temporal formulas over terms. The syntax of the language is described by mutual recursion in Table 1

Table 1: Syntax of EpTemp

$\beta := p \,[\!]\, \neg\beta \,[\!]\, (\beta \Rightarrow \beta)$	basic formulae
$t := 0 \,[\!]\, 1 \,[\!]\, (\int \beta) \,[\!]\, (t + t) \,[\!]\, (t.t)$	probabilistic terms
$\delta := (\Box\beta) \,[\!]\, (t \leq t) \,[\!]\, (\sim\delta) \,[\!]\, (\delta \supset \delta)$	EPPL global formulae
$\delta_1 := (\Box\beta) \,[\!]\, (t \leq t) \,[\!]\, (\sim\delta_1) \,[\!]\, (\delta_1 \supset \delta_1) \,[\!]\, (\mathsf{EX}\delta_1) \,[\!]\, (\mathsf{AF}\delta_1) \,[\!]\, (\mathsf{E}[\delta_1 \mathsf{U} \delta_1])$	EpCTL global formulae
$\delta_2 := (\Box\beta) \,[\!]\, (t \leq t) \,[\!]\, (\sim\delta_2) \,[\!]\, (\delta_2 \supset \delta_2) \,[\!]\, (\mathsf{X}\delta_2) \,[\!]\, (\mathsf{F}\delta_2) \,[\!]\, (\delta_2 \mathsf{U} \delta_2)$	EpLTL global formulae

where $p \in \Lambda$.

In fact, the language presented here is the set of *Closed EpTemp formulae*. The original language is richer, allowing for algebraic real variables.

Classical abbreviations for propositional connectives ($\wedge, \vee, \Leftrightarrow$), comparison operators ($=, \geq, <, >$), algebraic numbers ($\frac{3}{4}, \sqrt{2}$), temporal connectives(AX,EF,AU) and even propositional analogous for global connectives ($\cup, \cap, \equiv$) are used freely.

In order to introduce EpTemp semantics, it is easier to define first EPPL semantics:

A model of EPPL is a tuple $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ where $(\Omega, \mathcal{F}, \mu)$ is a probability space and $\mathbf{X} = (X_p)_{p\in\Lambda}$ is a stochastic process over $(\Omega, \mathcal{F}, \mu)$ where each X_p is a Bernoulli random variable, that is, X_p ranges over $\mathbf{2} = \{0, 1\}$.

One can associate with each $\omega \in \Omega$ a valuation $v_\omega : \Lambda \to \mathbf{2}$ s.t. $v_\omega(p_i) = X_{p_i}(\omega)$.

Furthermore, each basic formula induces a Bernoulli random variable $X_\beta : \Omega \to \mathbf{2}$:

- $X_{(\neg\beta)}(\omega) = 1 - X_\beta(\omega)$;
- $X_{(\beta_1 \Rightarrow \beta_2)}(\omega) = \max((1 - X_{\beta_1}(\omega)), X_{\beta_2}(\omega))$.

Given an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$, the semantics of global formulae is defined in the following way:

- Denotation of probabilistic terms:
  - $[\![0]\!]_m = 0$; $[\![1]\!]_m = 1$;
  - $[\![(t_1 + t_2)]\!]_m = [\![t_1]\!]_m + [\![t_2]\!]_m$; $[\![(t_1.t_2)]\!]_m = [\![t_1]\!]_m.[\![t_2]\!]_m$;
  - $[\![(\int \beta)]\!]_m = \mu(X_\beta^{-1}(1))$ is the probability of observing an outcome ω such that v_ω satisfies β.
- Satisfaction of EPPL global formulae:
  - $m \Vdash (\Box\beta)$ iff $\Omega = X_\beta^{-1}(1)$;
  - $m \Vdash (t_1 \leq t_2)$ iff $[\![t_1]\!]_m \leq [\![t_2]\!]_m$;
  - $m \Vdash (\sim\delta)$ iff $m \nVdash \delta$;
  - $m \Vdash (\delta_1 \supset \delta_2)$ iff $m \Vdash \delta_2$ or $m \nVdash \delta_1$.

We can now describe the semantics for EpTemp . A *probabilistic Kripke structure* is a tuple $M = (S, R, L)$ where S is a set of states, $R \subseteq S \times S$ is a total transition relation (called the *accessibility relation*) and L is a map from S to the set of EPPL models.

The satisfaction relation is defined by structural induction:

- For EpCTL:
  - $M, s \Vdash \Box\beta$ iff $L(s) \Vdash \Box\beta$;
  - $M, s \Vdash (t_1 \leq t_2)$ iff $L(s) \Vdash (t_1 \leq t_2)$;
  - $M, s \Vdash (\sim\delta)$ iff $M, s \nVdash \delta$;
  - $M, s \Vdash (\delta_1 \supset \delta_2)$ iff $M, s \nVdash \delta_1$ or $M, s \Vdash \delta_2$;
  - $M, s \Vdash (\mathsf{EX}\delta)$ iff $M, s' \Vdash \delta$ with $(s, s') \in R$;
  - $M, s \Vdash (\mathsf{AF}\delta)$ iff for all path π over R starting in s there exists $k \in \mathbb{N}$ such that $M, \pi_k, L \Vdash \delta$;
  - $M, s \Vdash (\mathsf{E}[\delta_1 \mathsf{U} \delta_2])$ iff there exists a path π over R starting in s and $k \in \mathbb{N}$ such that $M, \pi_k, \Vdash \delta_2$ and $M, \pi_i, \Vdash \delta_1$ for every $i < k$.
- For EpLTL:

Recall that a CTL or LTL model is a *Kripke Structure*, a tuple $M = (S, R, L)$ where S is the set of states, $R \subseteq S \times S$ is the accessibility relation and $L : S \to 2^{\Xi}$ maps each state to a valuation over Ξ.

For implementation purposes, we will consider Ω finite with, at most, one element for each valuation $v \in 2^{\Lambda}$. This is reasonable since the variables of $\mathbf{X}$ are "blind" to different elements of Ω that induce the same valuations. Since the association is now bijective, we will call *global valuations* to the elements of Ω. We will also assume that all elements in Ω have non-zero measure when seen as singleton sets (we just take out the zero-measure ones), meaning that $\Box\beta$ and $\int \beta = 1$ have the same semantics in this case.

Under these assumptions, knowledge of the joint distribution of the random variables X_p is enough to describe an EPPL model. A more detailed explanation of these procedures can be found herePLACEHOLDER.

Using the tool with the GUI

Just like classical model checking with CTL or LTL consists in, given an adequate formula and a Kripke Structure, determining if the structure satisfies or not the formula, model checking EpTemp consists in, given an adequate temporal formula and a Probabilistic Kripke Strucutre (an extension of Kripke Structures), determining if the structure satisfies or not the formula.

In order to use the tool as a blackbox, you will first need to describe the states of the Probabilistic Kripke Structure. Under our assumptions, only the description of the joint distribution of the random variables associated with the propositional symbols of Λ is needed. This is done through the *factorization* of the joint distribution and posterior encoding in a *Bayesian Network*. It is simpler than it sounds.

The states are represented by .mbx files. These are produced with MSBNX, a free Microsoft program to describe Bayesian Networks (BN's). For simplicity,

we will use the same name for a propositional symbol and it's associated random variable.

Each BN must have one variable for each propositional symbol that appears on the formula that we want to check, but may have any number of spurious variables.

Note: At this time, the tool does not have a syntax checker and it is common for one to declare a variable with a slightly different name in the formula and in the state, or even to forget to declare a variable in the states at all! A lot of time was lost searching for a bug that turned out to be a "x1" in the formula and a "X1" in the state. We plan to develop said syntax checker, but for now, you should be careful when naming state variables.

After you have created and saved all states of the structure, you must change their extensions from .mbx to .txt and move them to the directory where you have unpackaged the tool (a browsing function will also eventually be implemented). Now, we just run the "EpTemp.exe" file to start the GUI.

The GUI has a usual wizard layout. The first panel lets you choose if you want to model check EpCTLor EpLTL. Just select the option you want.

The second panel lets you choose the EpTemp formula you want to check. You can either use a formula you have created previously or create a new formula. If you choose to use an old formula, just check the first option and input the name of the formula you have previously created (using this tool).

If you choose to create a new formula, check the second option. This will enable the global connectives panel. Introduction of formulae is done in prefix notation, although the on-screen presentation of said formulae is the usual.

This means that, in order to check the formula $AF((\Box(x_1 \wedge x_2)) \cap (\int x_2 = \frac{3}{4}))$ you should create x_1 and x_2 (write "x1" or "x2" on the "New prop. Var." and "Create"), click AF, $\cap$, $\Box$, $\wedge$, select "x1" in the "Propositional Con." dropbox, click "Insert", select "x2", click "Insert", $=$, $\int$, select "x2", click "Insert", enter "0.75" in the box in "Term Con." and click "Insert".

While you are inserting the formula, it appears in the white bar above the buttons. Five dots show the spot where you are currently inserting the formula while an underscore denotes spots you still have to fill. You can always save a new formula for later use by checking the "Save formula" checkbox and choosing a name for it. Be cautions when doing this as you can accidentally overwrite a previous formula with the same name.

The third panel is used to define the Probabilistic Kripke Strucutre. First, we introduce the names of each file with a state description, without extensions (the name of the file will be the name of the state). Click "Insert" to add each state to the list of states. Then, we select the initial state in the dropbox. When all states are introduced, click "Start defining the transition relation".

The transition relation is defined by stating, for each state, which states are seen by it.

The state for which you are currently declaring allowed transitions will be the one whose name is on the top box. For each state it can see, select the seen state in the dropbox and "Add" it. When you have exhausted all allowed transitions (you must have at least one to unlock "Next state", as the transition relation is

total), just press "Next state" and proceed to the following declared state. When you finish this procedure, just press "Finish" button (which appears where "Next State" previously was).

When you click "Next", since all the necessary information has been inputed, the kernel is called and the EPPL computations are performed. NuSMV is then called and the output is interpreted.

The fourth and last panel just presents the output.

Using the tool with a scripted input file

Having a GUI is handy if one wants to check a simple academic example or find a minimal counter-example, but most real-world uses of a model checker are far to complex to input every single state and transition by hand. If one finds him or herself in this situation, EpTempMC allows him or her to forfeit the GUI module and call the kernel directly, writing the input in a .txt file.

In order to write the input, create a .txt file named "inputfile.txt". The structure of the file is best explained by an example, as in figure PLACEHOLDER

Formulae codifications are just the way the GUI represents formulae. Each connective has a codification to an integer number and variables and terms are preceded by their own identifier. The codification of the formula is just the concatenation of all these individual codifications.

Table 2 describes the codification of each connective, but a much easier way of getting the code is to run the GUI, input the formula, save it for later use and then abort the GUI procedure. This leaves a "formulaname.ctl" or "formulaname.ltl" file behind with the codification written on it. Just open the file with any text editor and copy-paste the string in the input file to save a lot of trouble.

The names of the states must coincide with the name of the file where the state is described, sans extension. Identification of states must be biunivocal, so repeated names will generate errors.

After listing all states and before declaring the first state, a line containing a single "?" must appear. "?" was chosen completely at random to be used as a separator. The first state must, of course, be one of the declared states.

After the initial state has been chosen, the declaration of the transition relation is done by listing, for each state, which states are seen by it. Therefore, for each state, there must be a block consisting of, in this order:

- one line with the separator "?";
- one line with the name of the parent state;
- one line with the name of each state seen by the parent state;
- one blank line;

After all transitions have been declared, another separator, "}" must be placed to signal the end of the file.

This input file should be placed in the same folder as EpTempMC , and then you just need to run the kernel and read the output from "outputfile.txt". This simple structure of the input file allows for one to script the description of a complex structure to use the MC.

Annexes

Setting the Path

- In Vista:
  - On the “Start” menu, right-click “Computer” and choose “Properties”;
  - On the left, go to “Advanced System Properties”;
  - In the “Advanced” tab, click “Environment Variables”;
  - In the “System variables” window, highlight “Path” and click “Edit...”;
  - Concatenate the path you want to add and click “Ok” (the concatenation symbol in windows is “;”);
- In Xp/2000:
  - On the desktop, right-click My Computer and choose “Properties”;
  - In the “Advanced” tab, click “Environment Variables”;
  - In the “System variables” window, highlight “Path” and click “Edit...”;
  - Concatenate the path you want to add and click “Ok” (the concatenation symbol in windows is “;”);

Example of interface window with non-ASCII characters

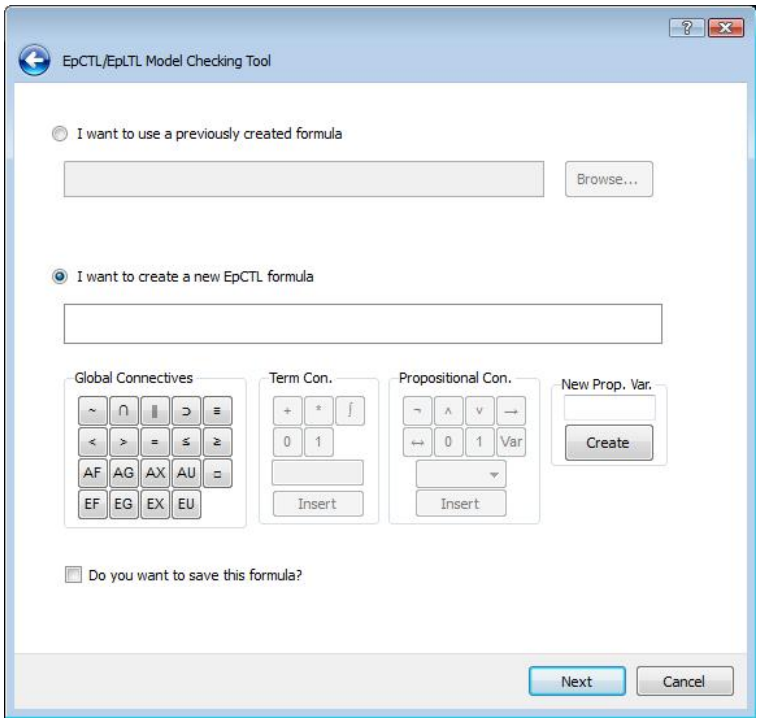

Figure 1: Interface window with non-ASCII characters

Correspondence between syntaxes and internal representation

EPPL *Symbol*	ASCII Symbol	Internal Representation
$\cap$	&&	-1
$\cup$	||	-2
$\supset$	=>	-3
$\equiv$	<=>	-4
$\sim$	!	-5
$\Box$	#	-6
$\int$	$	-7
$<$	<	-8
$>$	>	-9
$\leq$	=<	-10
$\geq$	>=	-11
$=$	=	-12
$\wedge$	&	-13
$\vee$	|	-14
$\Rightarrow$	->	-15
$\Leftrightarrow$	<->	-16
$\neg$	~	-17
0 (propositional)	0	-18
1 (propositional)	1	-19
0 (term)	0	-20
1 (term)	1	-21
+	+	-22
.	*	-23
AX	AX	-24
AG	AG	-25
AF	AF	-26
A_U_	A_U_	-27
EX	EX	-28
EG	EG	-29
EF	EF	-30
E_U_	E_U_	-31
X	X	-32
G	G	-33
F	F	-34
U	U	-35
term	*value*	-40 value
variable	*name*	1 name

Table 2: Correspondence between syntaxes and internal representation