Documentation of EMPEPA

Nil Geisweiller

August 25, 2006

Contents

Chapter 1

Introduction

EMPEPA is a tool that permits to find the value inside a PEPA model (Hillston 1994, 1996) according to a set of observations obtained, typically, by observing the real system. It implements two algorithms (Geisweiller 2005, 2006) both based on the EM *(Expectation-Maximization)* method (Dempster et al. 1997 and Wu 1983). The first algorithm takes in input a finite set of absorption durations :

$$14.2,\ 10.2,\ 7.3,\ 5.8,\ 11.4,\ldots$$

The second algorithm takes in input a finite set of *partially observable executions* :

$$\begin{array}{l} init \xrightarrow{t_1} a_1 \ldots a_{m-1} \xrightarrow{t_m} stop \\ \vdots \end{array}$$

where a_i is the i^{th} observed action and t_i is the spent time between the two observed actions a_{i-1} and a_i.

Chapter 2

Installation

2.1 Where to find EMPEPA

`http://sourceforge.net/projects/empepa`

2.2 Where to download EMPEPA

`http://sourceforge.net/project/showfiles.php?group_id=143706`

2.3 How to install EMPEPA

First, make sure you have GSL (Gnu Scientific Library) and GSLCBLAS installed on your system (visit http://www.gnu.org.software/gsl for more information). Untar EMPEPA where you like :
`tar xvzf empepa-0.9.2.tar.gz`
tar creates a directory (`empepa-0.9.2` for the version 0.9.2), so go to this directory :
`cd empepa-0.9.2`
then type : `./configure`
or
`./configure --prefix=WHERE_YOU_WANT`
to put the project where you want. Then type :
`make`
and finally :
`make install`
This last command may require the root privilege.

If you do not succeed to install EMPEPA or if you find a bug please send me an e-mail at the address `a-lin@users.sourceforge.net` mentioning EMPEPA into the title of the e-mail.

Chapter 3

How to use EMPEPA

The different functions of EMPEPA are invoked with the following command line :

`empepa [options] file1 [file2]`

The program EMPEPA always asks at least one file in input, `file1`, that contains a PEPA model, whose syntax is described at *Section* 4.1. Depending on the selected options it expects in input a second file, `file2`, describing the observed data, whose syntax is described at *Section* 4.2. The options are the followings :

`-?` or `--help`

Display some help on how to use EMPEPA.

`-a ALGO` or `--algo=ALGO`

where `ALGO` is the resolution scheme of the ordinary differential equation systems used by the algorithms. `ALGO` can be one the following key words :

1. `rk2` for Runge-Kutta (2,3),
2. `rk4` for Runge-Kutta of the 4th order
3. `rkf45` for Runge-Kutta-Fehlberg (4,5),
4. `rkck` for Runge-Kutta Cash-Karp (4,5),
5. `rk8pd` for Runge-Kutta Prince-Dormand (8,9),
6. `rk2imp` for implicit Runge-Kutta with Gaussien points of order 2,
7. `rk4imp` for implicit Runge-Kutta with Gaussien points of order 4,
8. `gear1` for M=1 implicit Gear method,
9. `gear2` for M=2 implicit Gear method,
10. `simul` for an experimental approximation method using simulations.

By default the selected scheme is `rk4`.

`-r POSITIVE_REAL` or `--resolution=POSITIVE_REAL`

where POSITIVE_REAL is a positive real corresponding to the precision of the chosen resolution method. By default this value is 1e-12.

`-i ITERATIONS` or `--iterations=ITERATION`

where ITERATIONS is the number of iterations of the algorithm. By default this number is 10.

`-s` or `--simulate`

This option permits to use EMPEPA in a simulation mode. In this case no file `file2` is expected in the command line. However, if this option is added then EMPEPA expects the following options :

`-o DATA_FILE` or `--output=DATA_FILE`

where DATA_FILE is the name of the file where EMPEPA has to write the observations obtained by simulations. If the option :

`-t STOP_RATE` or `--stoprate=STOPE_RATE`

is added then EMPEPA writes in DATA_FILE a list of partially observable executions with the average duration of each execution 1/STOP_RATE. Otherwise EMPEPA writes absorption durations (beware, in this case the PEPA model has to be absorbing). The following option :

`-n NBR_SIMUL` or `--numsimul=NBR_SIMUL`

defines the number of observations to write in the file DATA_FILE, that is the number of absorption durations or the number of partially observable executions. If EMPEPA is used to maximize the observations and that the solving method is `simul` then NBR_SIMUL designs the minimum number of simulations used to make the approximation of the expectations.

Above is an example of a command line which launches EMPEPA with the numerical method Runge-Kutta Prince-Dormand (8,9) with the precision 10^{-10} to maximize the likelihood of the observations contained into the file `observation.data` and considering the PEPA model described by the file `model.pepa` :

```
empepa -a rk8pd -r 1e-10 observations.data model.pepa
```

The next example permit to get 100 partially observable executions with stop rate 0.01 by simulating the PEPA model in the file `model.pepa`. The observations are written in the file `observations.data` :

```
empepa -s -n 100 -t 0.01 -o observations.data model.pepa
```

Chapter 4

Syntax of PEPA models and observations

The non-terminals are indicated in italic font whereas the terminals are indicated in courier font. The tokens are indicated in capital italic character. The empty word is symbolized by ϵ. The two typed tokens have been used : *ID* representing any identifier and *PR* representing any positive real.

$$ID \quad ::= \quad [\texttt{A..z}]([\texttt{A..z}]|\_)^*$$

$$PR \quad ::= \quad [\texttt{0..9}]^+(.[\texttt{0..9}]^*|\epsilon)(\texttt{e}(\texttt{-}|\texttt{+})[\texttt{0..9}]^+|\epsilon)$$

4.1 Syntax of PEPA models

The syntax used to represent the PEPA model is described here. This syntax is similar to the one used in the model checker PRISM with the additional key word `nil` to designs the idle process and a list of declarations of variables valued or not.

PEPAModel ::= *VarDecl ConstDecl TermDecl RootTerm*

VarDecl ::= `var` *VarList* `;`

VarList ::= *VarList* `,` *Variable* | ϵ

Variable ::= *ID* | *ID* `=` *RP*

ConstDecl ::= `const` *ConstList* `;`

ConstList ::= *ConstList* `,` *ID* `=` *PR* | ϵ

TermDecl ::= *TermDecl* ♯ *ID* = *PEPATerm* ; | ϵ

PEPATerm ::= *PEPATerm* / { *IDList* } | *PEPATerm* + *PEPATerm*
| (*Action* , *Rate*) . *PEPATerm* | *ID* | nil | (*PEPATerm*)

Action ::= tau | *ID*

Rate ::= *PR* | *ID* | infty

RootTerm ::= *PEPATerm* | *RootTerm* < *IDList* > *RootTerm*
| (*RootTerm*)

IDList ::= *IDList* , *ID* | ϵ

Below is an example of a PEPA model described in this syntax :

```
var x=2.1e-3, y;

const z=4.2;

♯P1=(a,x).(b,z).P1+(c,y).P1;
♯P2=((b,x).(a,z).P2)/{b};

P1<a>P2
```

4.2 Syntax of observations

Here is presented the syntax of the observations. There are two possible types of observations : *AbsorptionDurations* represents a set of absorption durations and *PartObsExecList* represents a set of partially observable executions. The symbol $\hookleftarrow$ means one or several carriage return.

AbsorptionDurations ::= *AbsorptionDurations Abs* | *Abs*

Abs ::= *PR* $\hookleftarrow$ | *PR PR* $\hookleftarrow$

PartObsExecList ::= *PartObsExecList PartObsExec* | *PartObsExec*

PartObsExec ::= init *ObsTimeList* - *PR* > stop

ObsTimeList ::= *ObsTimeList* - *PR* > *ID* | ϵ

Below is an example of a set of 4 absorption durations. The second absorption time, 2.4e-1, has been observed twice and the absorption time 5.9 three times.

1.2e-2
2.4e-1 2
4.3
5.9 3

The next example corresponds to a list of 4 partially observable executions :

```
init -0.3>a -1.2>b -2.3e-1>stop
init -1.2>stop
init -2.1>c -3.1e-6>a -4.34>b -4.8>b -0.2>stop
init -1.1>c -2.7>a -3.3e-3>b -2.3>stop
```

Bibliography

Dempster and Laird and Rubin 1977, Maximum likelihood from incomplete data via the EM algorithm. Journal : J. Roy. Soc. Ser. B., volume 39, pages 1-38.

Geisweiller 2005, Fitting General Distributions within PEPA Models, in Proc. PASTA 2005, Edinburgh, Scotland and PMCCS 2005, Torino, Italia.

Geisweiller 2006, Étude sur la Modélisation et la Vérification Probabiliste d'Architectures de Simulations Distribuées pour l'Évaluation de Performances. PhD Thesis.

Geisweiller 2006, Finding the Most Likely Value inside a PEPA Model According to a Partially Observable Executions.

Hillston 1994, Compositional Markovian Modelling Using a Process Algebra. In: Stewart, W.J. (ed), Numerical Solution of Markov Chains. Kluwer.

Hillston 1996, A Compositional Approach to Performance Modelling. Cambridge University Press.

Wu 1983, On the convergence properties of the EM algorithm. In journal Ann. Statist., volume 11, pages 95-103.