A Tutorial in Using Ipc

Allan Clark, Jeremy Bradley and Stephen Gilmore

October 10, 2007

1 Introduction

PEPA[1] is a poplular stochastic process algebra. The ipc software tool is a compiler for PEPA models into a format suitable for analysis by the Hydra[2] tool. This tutorial will guide you through using the the ipc software tool to perform various measurements over various models. For full documentation on the ipc tool please refer to the manual.

2 The Basics

We begin with a basic model and will subsequently augment the model to take into account more interesting features of PEPA and to allow some more sophisticated measurements. Our model is intended to represent a typical workplace coffee-room. The first model is a straight-line model representing only a single tea-drinker. The concrete syntax source for this first model is given in Figure 1.

```
r_thirst = 0.01 ; r_boil =  1.0 ; r_pour   = 10.0 ;
r_milk   = 6.0 ; r_stir = 10.0  ; r_drink = 0.1    ;

TeaDrinker = (thirst      , r_thirst) . MakeTea    ;
MakeTea    = (boil_kettle, r_boil)    . PourWater  ;
PourWater  = (pour        , r_pour)   . AddMilk    ;
AddMilk    = (milk        , r_milk)   . Stir       ;
Stir       = (stir        , r_stir)   . Enjoy      ;
Enjoy      = (drink       , r_drink)  . TeaDrinker ;

TeaDrinker
```

Figure 1: The source for the simple tea drinker model.

The rates described in table 1.

The simplest invocation of ipc on this model would be the following command:

```
ipc simpleTea.pepa
```

However this will simply produce a Hydra (.mod) with no measurement specifications. Generally we wish to compute some performance measurement

Rate Name	Value	Meaning
r_thirst	0.01	Thirsts once every 100 minutes
r_boil	1.0	The kettle takes one minute to boil
r_pour	10.0	It takes 6 seconds to pour
r_milk	6.0	It takes 10 seconds to add the milk
r_stir	10.0	It takes 6 seconds to stir the tea
r_drink	0.1	It takes 10 minutes to drink the tea

Table 1: The meaning of the rates in the simpleTea PEPA model.

over our model. In `ipc` there are four general kinds of measurements one can specify.

- Steady-state
- Passage-time
- Transient
- Count measures

Each of these will be described in more detail as we apply each to the first simple model. Special probe components are used to specify complex performance measurements however to begin with the user need not know about these and use the simpler interface of specifying activities of interest. The first three kinds of measurements are require 'start' and 'stop' actions to be given by the user, while a count measure requires only one set of action names. We now describe each of these four kinds of measurements in detail.

2.1 Passage-time Measurements

A passage-time measurement is used to measure between two events. The user specifies a set of start actions, the observation of the model performing any one of these actions will start the measurement. The user also specifies a set of stop actions and the measurement is terminated when the model performs any actions within that set. Without further ado a possible passage-time measurement over our simple model would be to measure the passage between a *thirst* activity and a *stir* activity. This can be done with the following command line:

```
ipc --source thirst --target stir simpleTea.pepa
```

This will produce a `.mod` file suitable for analysis with Hydra however `ipc` also provides the `----run-hydra` flag which will do this for the user.

```
ipc --source thirst --target stir --run-hydra simpleTea.pepa
```

We will make extensive use of this flag during this tutorial. Having run this command-line there should now be a `simpleTea.PT_RESULTS` file. This can now be processed with `gnuplot`. This will produce a graph as shown in Figure 2. This graph shows the cummulative distribution function, this means it maps the probability that the given passage has completed at the point in time corresponding to the x-axis. In this particular case we can see that the probability rises quickly before flattening out close to probability 1 (on a long enough timeline all teas are made).

After about 50 seconds it has a 50% chance of having completed, so after 50 seconds there is an equal probability that the passage will have completed as there is that it has not completed. After about 3 minutes there is a high probability that the passage has completed.

Note that the measurement is from the completion of a *thirst* activity and the completion of a *stir* activity. Therefore in this measurement the time taking to perform the *thirst* activity is irrelevant but the time take to perform the *stir* activity does matter.

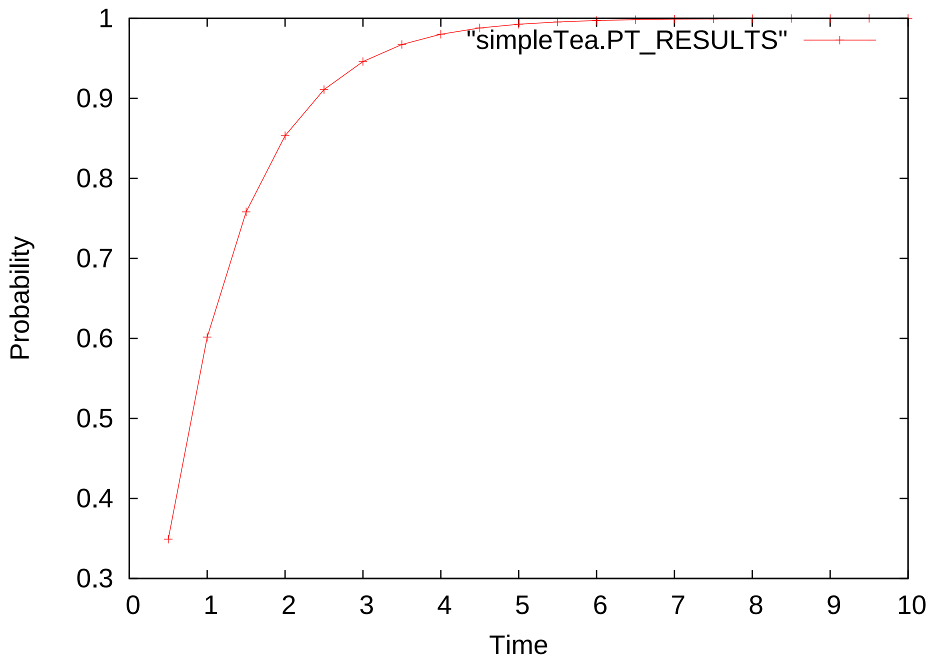

Figure 2: Shows the cummulative probability of completion against time.

2.2 Steady-state Measurements

A steady-state measurement analyses the long-term probability that the model is in a given state or set of states.

This first measurement analyses the probability that the *TeaDrinker* is currently in the process of making their tea. To do this we start the measurement at the *thirst* activity and stop at the *stir* activity. Shown here is both the command used to invoke `ipc` and the response from the subsequent run of Hydra. Note here the short forms of the `--source` and `--target` options have been used, these are `-s` and `-t` respectively.

```
 ipc -s thirst -t stir --steady --run-hydra simpleTea.pepa
```

```
State Measure 'steady_measure'

  mean                      1.2271774918e-02
```

This tells us that there is a very low probability, just over 1 percent, that the user is currently making tea, this is intuitively correct since generally the user is either enjoying a tea or not yet thirsty. We can therefore also measure the probability that the *TeaDrinker* has finished their last cup of tea but is not yet ready for another. To do this we measure the probability of being in any state between the occurrence of the *drink* activity and the occurrence of a *thirst* activity (in this case there is only one such state).

```
 ipc -s drink -t thirst --steady --run-hydra simpleTea.pepa
```

```
State Measure 'steady_measure'
  mean                  8.9793475007e-01
```

This tells us that there is almost an 81 percent chance that the user is in this state.

2.3 Transient Measurements

Transient measures are similar to passage time measurements but may be used to measure the short-term rather than steady-state probabilities. Our simple first model behaves similarly in the short-term than in the long-term this is because there are no interacting components so there is no need for the single straight line component to 'settle' into a steady-state. A transient measurement is invoked with the `----transient` flag, below is shown a sample command-line. As with a passage-time measurements the results are written to the file named `simpleTea.PT_RESULTS`.

```
 ipc -s thirst -t stir --transient --run-hydra simpleTea.pepa
```

2.4 Count Measurements

Count measures are used to calculate the mean rate at which a given set of activities occurs within a model. The mean rate an activity is performed depends upon the mean rate at which the activity is enabled and the long-term probability that the model is in a state in which the activity is enabled. For our simple model we could calculate the mean rate at which the *TeaDrinker* has a cup. This may be calculated with the following command-line:

```
 ipc --count-measure drink --run-hydra simpleTea.pepa
Count Measure 'my_count_measure' mean 0.008979347501
```

These results tell us that it is quite a low rate in comparison to the rates involved in making the tea, again as one would expect. At a mean rate of approximately 0.009 this means the tea-drinker enjoys a cup every 1/0.009 minutes, or once every 111 minutes. This in intuitively correct since the tea drinker 'thirsts' at a rate of once every 100 minutes and the time taken to make and drink the tea are both small by comparison.

3 Intermediate

We will now consider some more complex measurements. For this we update our model to include a boiler which continuously boils water for the tea drinkers.

The model used for these measurements is given in Figure: 3.

```
r_thirst = 0.01 ;
r_milk   = 6.0 ; r_stir = 10.0  ; r_drink = 0.1   ;

TeaDrinker = (thirst     , r_thirst) . PourWater  ;
PourWater  = (pour       , infty )   . AddMilk    ;
AddMilk    = (milk       , r_milk)   . Stir       ;
Stir       = (stir       , r_stir)   . Enjoy      ;
Enjoy      = (drink      , r_drink)  . TeaDrinker ;

r_pour   = 10.0 ; r_cool   = 0.2 ;
r_boil   =  1.0 ; r_refill = 10.0 ;

Boiler       = (cool, r_cool) . CoolBoiler
             + (pour, r_pour) . RefillBoiler
             ;
CoolBoiler   = (boil, r_boil)     . Boiler     ;
RefillBoiler = (refill, r_refill) . CoolBoiler ;


TeaDrinker < pour > Boiler
```

Figure 3: The source for the tea drinker and boiler.

3.1 Probe Specifications

Rather than specifying start and stop actions one can specify a probe. We begin with an example first. Suppose we wish to find out the probability that the boiler has boiled three times in a row without any water being taken from it. If this probability is high then we can conclude that the boiler is inefficient since it will unnecessarily boil water which is not then subsequently used.

The probe specification to ask this question is the following: $((boil, boil, boil)/pour) : start, pour : stop$ This probe specifies that once we have witnessed three *boil* activities without witnessing a *pour* activity then the measurement should *start*. Once the measurement is started the a *pour* activity will stop it. We could perform a passage-time measurement on this model and probe however this would tell us only the expected time to a *pour* activity after witnessing a sequence of three *boil* activities. This is not what we want. Instead we use a steady-state measure to calculate the probablity that we are in any of the measured states. Therefore our command-line is:

```
 ipc --probe "((boil, boil, boil)/pour) :  start, pour :  stop" \
            --steady --run-hydra boiler.pepa
```

References

[1] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[2] J.T. Bradley, N.J. Dingle, S.T. Gilmore, and W.J. Knottenbelt. Extracting passage times from PEPA models with the HYDRA tool: A case study.

In S. Jarvis, editor, *Proceedings of the Nineteenth annual UK Performance Engineering Workshop*, pages 79–90, University of Warwick, July 2003.