The Ipc Manual

Allan Clark

October 10, 2007

1 Introduction

PEPA[1] is a poplular stochastic process algebra.

About this document ...

This document is a manual for users of the Imperial Pepa Compiler or ipc. Users are expected to already know how to interpret their models. A companion document, "The
ipc
tutorial" is available. The tutorial explains how to use
pepa
for stochastic modelling.

2 Usage

2.1 Trivial Invocation Options

These options only apply to the trivial invocations of `ipc`.

-v, `--version` The command-line:

$ ipc `--version`

will print the version of the ipc compiler and exit.

-h, `--help` The command-line

$ ipc `-- help`

will print an option and usage summary and exit.

2.2 Output Re-directing Options

By default the `ipc` command will generate output in the same directory and with a file name based on the input source name The options in this section allow the user to redirect the output from the `ipc` compiler

`--mod-file` FILE Sets the output `.mod` file which will be the input to the run of Hydra

`--output` FILE A generic output flag, this works regardless of the kind output that `ipc` is set to produce. For example instead of the default Hydra the compiler may have been set to produce a prism model file. The `--output` can be used to set the output file name

`--stdout` When debugging the compiler it can often be useful for the output to be redirected to the terminal for immediate inspection by the programmer. The `--stdout` sets the output file to be the standard out. This may also prove useful for piping the output into further processing tools.

2.3 Static Analysis Options

These options control the way that static analysis is performed.

`--staunch` The flag `-- staunch` allows the compiler to ignore warnings. By default this flag is off and when `ipc` performs any static-analys is over the input PEPA model if there are any warnings it will cause the compiler to cancel compilation. This behaviour can be suppressed with the `-- staunch` flag. This will cause the warnings to still be emitted but compilation will proceed anyway.

`--no-static-analysis` The flag `-- no-static-analysis` causes `ipc` to avoid performing the static analysis over the PEPA model. It will therefore produce no warnings or errors. Because of this compilation may fail mysteriously and hence the user is advised only to use this flag if they know exactly what they are doing and expect their model to fail static-analysis for some reason but wish to proceed to compilation anyway.

2.4 Probe Specification Options

This section describes the options which control the performance measure specification probes which are added to the model.

-p, `--probe` **PROBESPEC** The flag `-- probe` with the shortened version `p` is used to give a full probe specification.

`--no-master` The flag `-- no-master` is used to specify that no master probe should be automatically added to the model.

-s, `--source` **ACTIONS** The flags `-- source` and `-- target` with the short versions `s` and `t` respectively, set the state switching actions used in the master probe. If other probes are added using the `-- probe` option then they may perform immediate communication actions which are specified in the source and target action list. Both the `-- source` and `-- target` flags accept as argument a comma separated list of action names.

-t, `--target` **ACTIONS** see `--source`

2.5 Running Hydra

This section details options used for running the hydra tool after processing of the model by `ipc`.

`--run-hydra` The flag `--run-hydra` causes `ipc` to automatically run the hydra tool on the produced `.mod` file.

`--hydra` **PATH** If the hydra tool is not installed in a standard location the path to the Hydra executable can be given as an argument to the `--hydra` option.

2.6 Miscelleanous Options

This section describes options which don't fit under any of the previous subsections.

-V, `--verbose` No manual entry but the usage information states: logfile output to STDOUT

`--passage` No manual entry but the usage information states: perform a passage-time performance measurement (default)

`--steady` No manual entry but the usage information states: perform a steady-state measurement

`--transient` No manual entry but the usage information states: perform a transient analysis measurement

`--count-measure` **ACTIONS** No manual entry but the usage information states: perform a count measure over the given actions

`--no-measurement` No manual entry but the usage information states: do not output a performance measurement specification

`--steady-mean` No manual entry but the usage information states: use the 'mean' estimator in a steady-state measure

`--steady-variance` No manual entry but the usage information states: use the 'variance' estimator in a steady-state measure

`--steady-stddev` No manual entry but the usage information states: use the 'stddev' estimator in a steady-state measure

`--steady-distrib` No manual entry but the usage information states: use the 'distribution' estimator in a steady-state measure

`--start-time` **TIME** No manual entry but the usage information states: specify a time at which to start a performance measure eg passage-time

`--stop-time` **TIME** No manual entry but the usage information states: specify a time at which to stop a performance measure eg passage-time

`--time-step` **TIME** No manual entry but the usage information states: specify a the time steps for a performance measurement

`--solver` **SOLVER** No manual entry but the usage information states: specify which solution method to use/specify to hydra

`--rate` **DOUBLE** No manual entry but the usage information states: Override/specify a rate value on the command-line

`--rename-proc` **P=s** No manual entry but the usage information states: cause a renaming on the given process within the model

`--rename-rate` **r=s** No manual entry but the usage information states: cause a renaming on the given rate within the model

`--prioritise` **ACTIONS** No manual entry but the usage information states: increase the priority of the given actions

`--no-reduce-rate-exps` No manual entry but the usage information states: do not reduce the rate expressions

`--hide-non-coop` No manual entry but the usage information states: hide any activities which a component performs but does not cooperate over

`--process-num` **NUM** No manual entry but the usage information states: provide a process number which is used to select rates and processes

`--show-simplified` No manual entry but the usage information states: Show the simplified model in the comments of the compiled file

`--fsp` `--fsp` this is an experimental option to produce an LTSA model. This should not be considered working

2.7 Using ipc to evaluate a single model

In this section we will learn how to invoke ipc to peform various analyses over simple models.

As a simple example consider the PEPA model shown here in the concrete syntax of ipc.

```
r1 = 1.0;
r2 = 2.0;
r3 = 2 * r1;

P1 = (start, r1).P2;
P2 = (run, r2).P3;
P3 = (stop, r3).P1;

P1 <> P1
```

2.8 Basic Invocation

Invoking the compiler to give basic information can be done by specifying either the `--version` or `--help` flags.

```
ipc --version

ipc --help
```

Many of the flags and options have a corresponding one letter alias, for example the version can be printed with the command:

```
ipc -v
```

For the remainder of this manual we will use the long versions for clarity. The short versions can be found by reading the output of the help flag.

Beyond the trivial invocations involving version and help information, ipc must be invoked with a PEPA model argument. The general form is

```
ipc [flags and options] tiny.pepa
```

The simplest command-line for our `tiny.pepa` file would be:

```
ipc tiny.pepa
```

Such a simple command will produce a `tiny.mod` file suitable for analysis with Hydra. However it is generally more useful to provide some kind of measurement specification. Generally of course the point of solving the model is to make some measurement of the model. The following section 2.8.1 begins with the simplest way in which to specify a measurement.

2.8.1 Specifying source and target options

The ipc/Hydra tool chain can be used to calculate three major types of performance measure:

1. Passage time quantiles and distributions

2. Transient measures

3. Steady state measures

There are two ways in which to specify the states in which the measurement is concerned. In this section simple measurements which can be done by specifying sets of source and target actions are considered. Section 3 is concerned with more complex measurements.

As the names suggest, the source activity is used to start the measurement and the target activity is used to end the measurement.

Thus if we were interested in computing the passage from an occurrence of the `start` activity to the occurrence of the `stop` activity by either of the copies of the process in the tiny model then we would use the following command

```
ipc --source start --target stop tiny.pepa
```

This will generate a Hydra model file (`tiny.mod`) to be analysed by a subsequent run of Hydra. With each of these arguments more than one action may be given as a comma separated list of actions. For example one might give a command such as:

```
ipc --source halfFull,halfEmpty --target full,empty optimism.pepa
```

Since Hydra will more often than not be invoked on the output `.mod` file one can specify the `--run-hydra` flag. By default the kind of measurement is a passage-time measurement and `ipc` knows this and will run the corresponding `hydra-uniform` command after the initial run of Hydra itself. So by issuing the command:

```
ipc tiny.pepa --source start --target stop --run-hydra
```

You should produce the file: `tiny.PT_RESULTS`. The `gnuplot` program can then be used to produce a graph of the results such as the one in Figure 1. This graph plots the cummulative probability of completion of the passage against time. That is the probability that t seconds after observing a *start* activity a *stop* activity is observed.

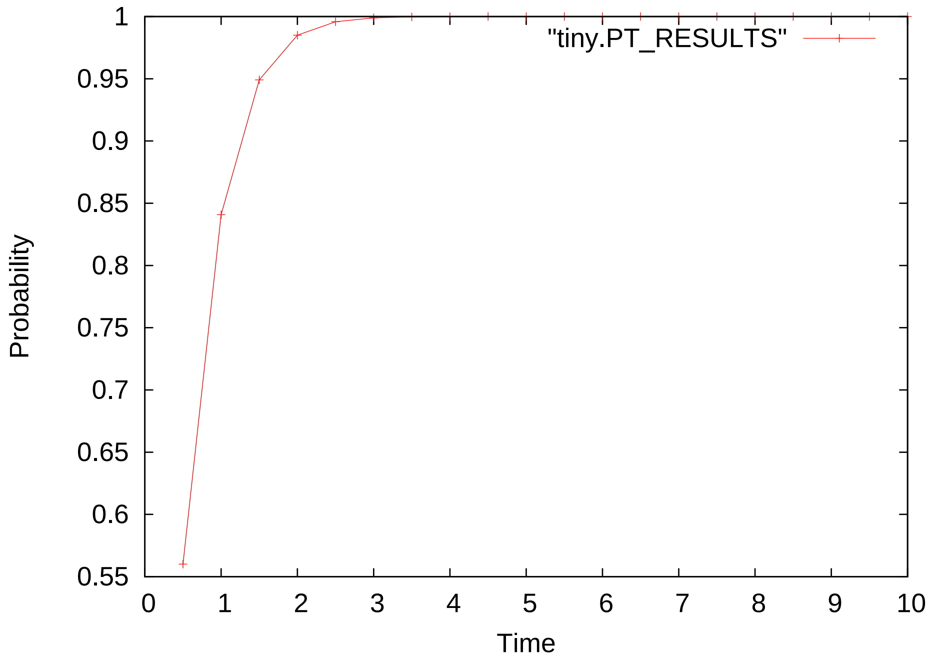
2.9 Changing the analysis type

We can specify the three different kinds of analysis with the three flags:

- `--steady`
- `--transient`
- `--passage`
- `--no-measurement`

A passage-time measure is the default unless the user does not specify either a measurement probe or a set of source/target actions. In this case `ipc` defaults to no measurement.

(¿¿ TODO: Explain what each of the measurement kinds mean and in particular for the given example what it would compute ??) .



htb

Figure 1: Shows the cummulative probability of completion against time.

2.10 Re-directing the output

The output model used as input to the Hydra tool will by default be placed in the file `tiny.mod`, but this can be overridden with the command line option `--mod-file` as in:

```
ipc --source start --target stop --mod-file ipc.mod tiny.pepa
```

Note that as mentioned above we will be giving command line options in their long forms, but most of them have short forms. The `--start` and `--target` options have the short forms `-s` and `-t` respectively. Because start and stop actions must be given for almost all runs of `ipc` we will use the short options to reduce the length of command lines where appropriate in the remainder of this manual.

2.11 Specifying start and stop times

Via Hydra, `ipc` computes performance measures such as probability density functions (PDFs) and cumulative distribution functions (CDFs). Together the PDF and CDF give a complete description of the probability distribution of a random variable. Both are evaluated against increasing time in order to produce function plots. `ipc` has default values for the start time and stop time for the plot, as well as the time step which determines how many evaluations of these functions Hydra is to do. The command-line options `--start-time`, `--stop-time` and `--time-step` allow us to override these defaults.

To specify a start time of 50 seconds and an end time of 100 seconds with a timestep of 10 seconds we would pass the following command-line options to ipc.

```
ipc --starttime 50 --stoptime 100 --timestep 10 ...
```

3 Probe Specifications

4 Attaching Performance Measurement Probes

References

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