

# Specification and Efficient Computation of Passage-Time Distributions in GPA

Matej Kohut    Anton Stefanek    Richard A. Hayden    Jeremy T. Bradley  
Department of Computing, Imperial College London  
{mk508,as1005,rh,jb}@doc.ic.ac.uk

**Abstract**—We present a significant extension to the *Grouped PEPA Analyser* (GPA) tool. We have augmented the tool with the ability to specify complex passage-time distributions with the *Unified Stochastic Probes* formalism and implemented efficient fluid analysis techniques to compute the distributions. The extension incorporates immediate signalling and weighted passive rates and permits two classes of passage time, namely *global* and *individual* passage times, to be computed.

We summarise how the different classes of passage-time query can be expressed using the *Unified Stochastic Probe* formalism and present some results from probed GPA models.

## I. INTRODUCTION

Fluid analysis or mean-field techniques, e.g. [1], allow us to analyse stochastic systems with large populations of identically behaved components. These techniques enable the study of increasingly more complex behaviour described in a variety of formalisms such as process algebras, Petri nets or systems of chemical equations. Traditionally, these give access to the time evolution of means and higher moments of populations of individual component types. However, various derived metrics are often needed, such as the distribution of the time it takes for the system to exhibit a desired observable sequence of actions. In the field of performance analysis, this can be useful when guaranteeing performance or reliability of a system. In particular *Service Level Agreements* (SLA) can be established as a contract between a service supplier and each individual client. For example, an agreement can state that “each client receives a service within  $60ms$  at least 99% of the time”.

The recent work of Hayden *et al.* [2] describes the *Unified Stochastic Probes* language that allows specification of complex passage-time queries. These can be defined outside of the main behavioural description and then used to produce an extended model. The results from the fluid analysis of this model are transformed to provide passage-time distributions in the original model. This method can quickly check whether the system satisfies a given SLA and can be further used in optimisation frameworks that help with the design of efficient systems.

The probe language allows a concise and user-friendly specification of complex behaviour-based queries. However, the translation to the extended model and the computation of the distribution is too complex to be performed by hand. In this paper, we introduce an extension to the *Grouped PEPA Analyser* (GPA) tool [3] that, for the first time, accepts a sophisticated passage-time query specification, automatically produces extended probed models and then applies fluid

analysis techniques to produce the resulting passage-time distributions.

In the following section we describe the syntax added to the tool and the individual improvements to the fluid analysis techniques of GPA that were necessary to fully automate the above method. We illustrate the tool on examples that had to be up until now hand-crafted. We believe that the combination of user-friendly model specification and efficient fluid analysis computation have significant potential for these techniques to be applied in practice.

## II. UNIFIED STOCHASTIC PROBES

The *Unified Stochastic Probes* formalism [2], allows a specification of an observable behaviour of a model. This can be given as a combination of regular expressions that accept sequences of actions exhibited by individual system components as well as logical predicates on the global model state space. Crucially, the resulting *probe* can be attached to the original model without changing the model definition. The fluid techniques then give access to the cumulative density function (CDF) of the distribution of the time it takes for the model to fully complete the specified behaviour.

### A. Example

We demonstrate the new features of GPA on a simple producer/consumer model defined in the GPEPA process algebra [4]. The system consists of a large number of producers and consumers. Each consumer can synchronise with a producer to obtain some data. Producers have a buffer that can become full, requiring a reset. The model can be defined in the original GPA syntax:

---

```
Consumer      = (think, rt).Consumer_get;  
Consumer_get  = (get_product, rg).Consumer_use;  
Consumer_use  = (use, ru).Consumer  
Terminal      = (setup, rs).Terminal;  
Terminal_get  = (get_product, T).Terminal  
              + (timeout, rti).Terminal;  
Producer      = (init, ri).Producer_ready;  
Producer_ready = (produce, rp).Producer_done;  
Producer_done = (get_rproduct, rgp).Producer  
              + (clear, rcl).Producer;  
Consumers{Consumer[N_c]}<get_product>  
Producers{(Producer <get_product> Terminal)[N_p]}
```

---

Formally, the system consists of  $N_c$  consumers and  $N_p$  producers with attached terminal components, synchronised on the `get_product` action.

In such a system, it would make sense to specify an SLA, for example, one that guarantees that each consumer will not

take longer than  $250ms$  at least 99% of the time to use two sets of data from the producers. This can be expressed by an individual steady state passage-time calculated by a global probe observing a local probe attached to a single consumer component [2]:

```
Probe ODEs(stopTime=250, stepSize=1, density=10)
  steady {
    GProbe = begin: start, end: stop <-
    observes {LProbe = think: begin, use[2]: end<- }
    where { Consumers{Consumer[N_c]} =>
      Consumers{(Consumer <think, use> LProbe)
        | Consumer[N_c - 1]} } }
```

The specification consists of 4 parts. An analysis is specified (using the standard GPA syntax) that will be used to calculate the passage-time distribution. The keyword `steady` determines that the steady state passage-time will be considered.

The body of the probe first contains a *global probe* definition that determines which pair of signals will be used as the `start` and `stop` of the passage-time. The `observes` block defines *local probes* that will observe individual components and report signals to the global probe. The `where` block defines how the local probes are attached to the system, in terms of a transformation of the model. In this example, we attach the probe to a single consumer component and leave the remaining  $N_c - 1$  consumer components unchanged.

To obtain the CDF of the passage-time computed from the first occurrence of the `start` signal instead of the steady state, the keyword `steady` can be replaced by the keyword `transient` and the repeating operators `<-` removed.

Another type of passage-time measure the tool supports is the *global passage time*. This is the time it takes until a *proportion* of given components in a group satisfy a probed behaviour. For example, we can look at the time until 30% of consumers consume 10 products:

```
Probe ODEs(stopTime=300, stepSize=1, density=10) {
  GProbe = eE: start, end[nc * 0.3]: stop
  observes { LProbe = get_product[10] : end }
  where { Consumers{Consumer[N_c]} =>
    Consumers{(Consumer<get_product> LProbe) [N_c]} } }
```

This is achieved by attaching the local probe to each consumer component and stopping the global probe only when the count of `end` signals reaches 30% of  $N_c$ . This command gives a point mass approximation to the passage-time when used with the `ODEs` analysis or empirical CDF when used with the `Simulation` analysis respectively.

Each command produces a graphical output of the CDF from each defined global probe and also exports the raw data for further processing. Figure 1 shows an example for the three passage-time classes. The transient example is for the passage-time until an individual producer empties its buffer.

### B. Implementation details

To support the complete probe syntax and related ODE analyses techniques, following features were introduced to GPA:

- Passive and weighted passive actions [2], to allow more general component specification and for direct translation of probes into GPEPA.

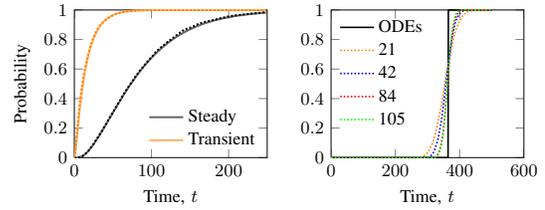


Fig. 1. Plots of individual passage time CDF (left) and global passage-time point mass approximation (right). The dotted lines are obtained from simulation. The global passage-time figure compares the point mass to the CDFs obtained from simulation of the model with increasing scale (the number shown is the value of  $N_c$ ).

- Immediate actions (signals) from iPEPA by implementing *vanishing state removal* [2]. Originally *well-behaved f-components* [2] require only deterministic initial behaviour, but we further restrict this to deterministic signalling paths for all states.
- Full set of Unified Stochastic Probe operations [2] along with the ODE based distribution computation for steady-state individual, transient individual and global passage time shown above. For comparison, these are also implemented in the built-in simulator in GPA.
- Originally, GPA dynamically generates Java classes for numerical solution to the underlying ODE systems. To support more complex models with larger component states, we added the option of dynamic generation of C++ code for the numerical computation.

### III. CONCLUSION & FUTURE WORK

We introduced an extension to the GPA tool that gives access to complex passage-time measures in large scale systems. The source code is available on the GPA website [code.google.com/p/gpanalyser](http://code.google.com/p/gpanalyser). We tested the techniques on a range of examples, including the large wireless sensor network case study in the original paper [2].

Apart from optimising probe translation algorithms, further improvements include full support for immediate signalling and passage-time computation using higher order moments. We plan to introduce a visual representation of probes and translation from *Performance Trees* [5], making the techniques even more accessible and potentially applicable in practice.

### REFERENCES

- [1] J. Hillston, “Fluid flow approximation of PEPA models,” in *QEST’05*, pp. 33–42, IEEE, Sept. 2005.
- [2] R. A. Hayden, J. T. Bradley, and A. Clark, “Performance specification and evaluation with Unified Stochastic Probes and fluid analysis,” *IEEE Transactions on Software Engineering*, Jan. 2012.
- [3] A. Stefanek, R. A. Hayden, and J. T. Bradley, “GPA - A tool for fluid scalability analysis of massively parallel systems,” in *QEST’11*, pp. 147–148, IEEE, Sept. 2011.
- [4] R. A. Hayden and J. T. Bradley, “A fluid analysis framework for a Markovian process algebra,” *Theoretical Computer Science*, vol. 411, pp. 2260–2297, May 2010.
- [5] T. Suto, J. Bradley, and W. Knottenbelt, “Performance Trees: A new approach to quantitative performance specification,” in *14th IEEE International Symposium on Modeling, Analysis, and Simulation*, pp. 303–313, IEEE, Sept. 2006.