SEMI-MARKOV PEPA: MODELLING WITH GENERALLY DISTRIBUTED ACTIONS

JEREMY T. BRADLEY

Department of Computing, Imperial College London, Huxley Building, 180 Queen's Gate, London SW7 2BZ
E-mail : jb@doc.ic.ac.uk

Abstract: Since the advent of Markovian Process Algebras, users have requested the ability to employ a greater variety of action distribution. We present a conservative extension to the popular Markovian process algebra, PEPA, which incorporates generally distributed sojourn-times for action duration. Just as a PEPA model generates a Markov chain for analysis purposes, so semi-Markov PEPA produces a semi-Markov chain. We discuss how semi-Markov PEPA models are analysed through Dingle and Knottenbelt's semi-Markov DNAmaca tool and present a small example for passage time analysis.

Key Words: Stochastic process algebra, Semi-Markov process, PEPA

1 INTRODUCTION

PEPA (Hillston 1996) is a Markovian process algebra that is easy to use and has a wide user community (*PEPA Resource Site* n.d.). It is perhaps unique among stochastic process algebras for this. It is a common request from PEPA users that the type of distribution that PEPA uses could be extended (Bowman, Bryans & Derrick 2001), so that models can be made more realistic.

PEPA is designed to be a Markovian process algebra which means that the duration of action events is specified by a sample from a random variable with an exponential distribution. What we have sought to do in this paper is allow generally distributed actions within a PEPA framework but in a way that preserves the ability to specify and analyse purely Markovian PEPA models. While it is undoubtedly useful for a stochastic process algebra to have greater distribution expressiveness, it is also desirable to keep the wealth of research that applies to Markovian PEPA intact.

The version of semi-Markov PEPA presented here extends a preliminary version (Bradley 2004) by also incorporating many priority levels of semi-Markovian and Markovian execution. In doing this, we have endeavoured to maintain the syntactic simplicity that underlies PEPA.

The question might reasonably be asked: why not simply replace the exponential distribution in PEPA with a general distribution? This was in fact one approach taken by Clark in the PEPAroni (Clark 1994) extension to PEPA. The reason for not doing this is that it generates an underlying Generalised Semi-Markov Process (GSMP) which is analytically intractable and can only be studied through simulation. Other stochastic process algebras which follow this approach include originally Strulo and Harrison's SPADES (Harrison & Strulo 1995), followed by D'Argenio's ♤ (D'Argenio, Katoen & Brinksma 1998, D'Argenio 1999), Bravetti et al's GSMPA (Bravetti, Bernardo & Gorrieri 1998) and Bravetti's iGSMPA (Bravetti & Gorrieri 1999).

Another instance where the distribution type has been extended in PEPA is PEPA$_{ph}^{\infty}$ (El-Rayes, Kwiatkowska & Norman 1999), where phase-type distributions are permitted for the action duration. The difficulty with using phase-types in a stochastic process algebra setting has always been that under cooperation the underlying Markovian state space undergoes an explosion in size (over and above the usual state-space explosion) equal to the product of the number of stages of phase-type in the cooperating actions. Added to this is the fact that at least 100 stages are needed to get an appropriate approximation of popular distributions such as the deterministic distribution. In this instance, even very simple cooperating models with small global state-spaces can end up with underlying Markov chains of many millions of states. On the other hand, as we discuss later, a phase-type process algebra is currently the only solution if full concurrency modelling is required with more general distribution types.

The attractive aspect to semi-Markov models is that performance results can be extracted analytically and tractably. Semi-Markov PEPA does not suffer from either the rare-event problems that are hard to pick up using simulations, or the distribution representation explosion of the phase-type solution.

While some SPAs have become *feature rich* and have intricate syntaxes to support such features, PEPA has successfully maintained a very small set of base operators. Thus, any attempt to modify or extend PEPA should hold to the original design qualities, especially that of syntactic simplicity. So our key design factors for semi-Markov PEPA or SM-PEPA are that it should: have Markovian PEPA as a subset; continue to have as simple a syntax as possible; have an analytically tractable stochastic model.

In Section 2.2, we look at what modelling advantages and applications we get from using a semi-Markov model. Section 3 defines the syntax for the extended process algebra. Finally, in Section 4, we illustrate the analysis of a semi-Markov PEPA model with a simple voting system.

2 SEMI-MARKOV MODELS

2.1 Background

Consider a Markov renewal process $\{(X_n, T_n) : n \geq 0\}$ where T_n is the time of the nth transition ($T_0 = 0$) and $X_n \in \mathcal{S}$ is the state at the nth transition. Let the kernel of this process be:

$$R(n,i,j,t) = \mathbb{P}(X_{n+1} = j, T_{n+1} - T_n \leq t \mid X_n = i) \quad (1)$$

for $i, j \in \mathcal{S}$. The continuous time semi-Markov process (SMP), $\{Z(t), t \geq 0\}$, defined by the kernel R, is related to the Markov renewal process by:

$$Z(t) = X_{N(t)} \quad (2)$$

where $N(t) = \max\{n : T_n \leq t\}$, i.e. the number of state transitions that have taken place by time t. Thus $Z(t)$ represents the state of the system at time t. We consider time-homogeneous SMPs, in which $R(n,i,j,t)$ is independent of any previous state except the last. Thus R becomes independent of n, so for any n:

$$\begin{aligned} R(i,j,t) &= \mathbb{P}(X_{n+1} = j, T_{n+1} - T_n \leq t \mid X_n = i) \\ &= p_{ij} H_{ij}(t) \end{aligned} \quad (3)$$

where $p_{ij} = \mathbb{P}(X_{n+1} = j \mid X_n = i)$ is the state transition probability between states i and j and $H_{ij}(t) = \mathbb{P}(T_{n+1} - T_n \leq t \mid X_{n+1} = j, X_n = i)$, is the sojourn time distribution in state i when the next state is j.

2.2 Semi-Markov Model Applications

Semi-Markov processes (Pyke 1961) are a low-level modelling formalism searching for a high-level description language. They provide the modeller with general distributions (rather than just exponential distributions) to describe atomic events however they supply no direct mapping for concurrent behaviour.

Existing concurrency-modelling paradigms, such as Petri nets and process algebras, map particularly well onto Markov models largely due to the *memoryless* nature of the exponential distribution. This property allows for the construction of an *expansion law* for concurrent Markov processes through an interleaving semantics.

Semi-Markov models have no such obvious mapping. SMPs have no easy interleaving semantics[1] that would allow the easy expression of the residual distribution for an interrupted action.

In order to define large SMPs, you do have to abstract yourself away from the transition matrix level. To this end, we invented SM-SPN—or semi-Markov Stochastic Petri Nets (Bradley, Dingle, Knottenbelt & Harrison 2003), an instantiation of ESPN (Dugan, Trivedi, Geist & Nicola 1984). In an SMP, for the process to select one of its possible successor states, a probabilistic choice is made and an arbitrary random delay is observed. In the same way, in an SM-SPN, a probabilistic choice is made between concurrently enabled transitions (based on pre-assigned weights) and an arbitrary random firing delay is observed. There is no concurrent running of transition clocks—only a single transition clock runs at a time.

To the best of our knowledge there is no equivalent semi-Markov style of stochastic process algebra although initial ideas behind adding general distributions to SPAs have been discussed in, for instance (Bradley 1999). Also an initial report on an early SM-PEPA design was produced as (Bradley 2003).

A possible argument for a lack of a semi-Markov SPA is: if only a single clock or distribution can be selected to run at a time in a semi-Markov model, what application can it have for modelling concurrent processes? We suggest two applications:

[1] SMPs that run concurrently are GSMPs and stochastic process algebra semantics do exist for GSMPs but they are not interleaving.

1. with a concurrent design but a single-threaded architecture on which to run it, e.g. a Java Run-time Environment on any single processor machine. The machine must time-slice between the different threads of the program: it must select a thread to run and then complete a certain amount of work on that thread before getting the chance to select a new thread to execute. This is ideally suited to the transition-selection style of semi-Markov processes; the selecting of a new thread being the probabilistic selection of a particular distribution.

2. for modelling mutual exclusion, e.g. failure–recovery modes, where normal concurrent operation is suspended and a single recovery process is repairing the system. This recognises the fact that the modeller will still want to have proper Markovian concurrency at their disposal. Markov processes are a strict subset of SMPs and thus it is possible to let the modeller have true concurrency when there are no general distributions enabled.

3 SEMI-MARKOV PEPA

3.1 Syntax

As before (Bradley 2004), the modifications to the syntax of PEPA are necessarily subtle in order to keep to our design criteria of having PEPA as an included subset. We keep unchanged choice, cooperation, hiding and the constant definitions.

Although we will need to address changes in semantic meaning for these operators for semi-Markov PEPA, our focus in this section lies in the change to the prefix operator $(a^{[n]}.D).P$. Here D is an arbitrary delay parameter for the action a, which can either take on the usual exponentially distributed delay or a weight, ω, and a generally distributed delay, defined by the Laplace transform, $L(s)$.

The syntax for a semi-Markov PEPA is as follows:

$$\begin{aligned} P \quad ::= \quad & (a^{[n]}, D).P \;\big|\; P + P \\ & \big|\; P \underset{L}{\bowtie} P \;\big|\; P/L \;\big|\; A \end{aligned} \qquad (4)$$

where:

$$D \quad ::= \quad \lambda \;\big|\; \omega : L(s) \qquad (5)$$

where λ is the standard PEPA exponential rate parameter:

$$\lambda \in \mathbb{R}^+ \cup \{r\top \mid r \in \mathbb{Q}, r > 0\}$$

3.2 Prioritised actions

The optional[2] numeric annotation, $[n]$, to the action label a denotes the priority of the action a and if omitted this defaults to an implicit value of 1. This priority augments the enabling rules of actions in a standard PEPA model: if an action a is enabled (by the standard PEPA definition) with priority n_1 and another action, b, is enabled with priority n_2, then the action a is only *priority enabled* and thus available for execution iff $n_1 \geq n_2$. Clearly, if no priorities are specified, the model returns to the standard PEPA enabling definition.

A further constraint on the model is applied saying that: a given priority level can consist solely of Markovian actions or solely of semi-Markovian actions. That is:

For a state P in a model, at a given n':

$$P \xrightarrow{(x^{[n']}, r)} P'$$

and $r \in \mathbb{R}^+ \cup \{r\top \mid r \in \mathbb{Q}, r > 0\}$, there exists no other state Q in the model which enables an action with priority $[n']$ and rate, $r \in \{\omega : L(s) \mid \omega \in \mathbb{R}, L(s) \in \mathsf{LAP}\}$.

There is a similar rule forbidding the enabling of Markovian transitions in semi-Markov priority levels:

For a state P in a model, at a given $n'' \neq n'$:

$$P \xrightarrow{(x^{[n'']}, r)} P'$$

and $r \in \{\omega : L(s) \mid \omega \in \mathbb{R}, L(s) \in \mathsf{LAP}\} \cup \{\top\}$, there exists no other state Q in the model which enables an action with priority $[n'']$ and rate, $r \in \mathbb{R}^+ \cup \{r\top \mid r \in \mathbb{Q}, r \neq 1\}$.

3.3 Apparent rate

A formal semantics for semi-Markov PEPA is presented at the end of the paper. Figure 4 preserves the standard Markovian PEPA behaviour, while Figure 5 adds those rules necessary to cope with the semi-Markov extensions to the formalism. One of the minor modification we have had to make to the standard PEPA definitions is in the apparent rate function from (Hillston 1996), usually ascribed $r_a(P)$, which describes the total observed rate of an action type a in a component P.

[2]This replaces the two-tier semi-Markov/Markov priority system of previous versions.

Here, we have had to introduce a *priority-aware* apparent rate function, $r_a^n(P)$, which as the name suggests measures only the total rate of an a action at priority level n in component P. This is used in Figure 4 where it applies to Markovian priority levels but has no meaning in semi-Markov execution. The priority-aware apparent rate function, $r_a^n(P)$, is defined by:

$$r_a^n(P) = \sum_{P \xrightarrow{(a^{[n]},\lambda_i)}} \lambda_i \qquad (6)$$

Other $\top$-rules, repeated here, required for the evaluation of the standard apparent rate function, are:

$$\begin{aligned} m\top < n\top &\quad : \quad \text{for } m < n \text{ and } m, n \in \mathbb{Q} \\ r < n\top &\quad : \quad \text{for all } r \in \mathbb{R}, n \in \mathbb{Q} \\ m\top + n\top = (m+n)\top &\quad : \quad m, n \in \mathbb{Q} \\ \frac{m\top}{n\top} = \frac{m}{n} &\quad : \quad m, n \in \mathbb{Q} \end{aligned}$$

Note that $(r + n\top)$ is undefined for all $r \in \mathbb{R}$ in Markovian PEPA therefore disallowing Markovian components which enable both active and passive actions in the same action type at the same time, e.g. $(a, \lambda).P + (a, \top).P'$.

3.4 Semi-Markov synchronisation

One interesting design decision is in the area of active synchronisation in a semi-Markov stochastic process algebra. This occurs when a model enables two a-actions in a semi-Markov priority level in components that cooperate over the action type a.

The same issue in Markovian process algebras created some discussion (Hillston 1994, Bradley & Davies 1999) over the underlying meaning of synchronisation. What PEPA emulates in its model of active synchronisation is a last-to-finish style of synchronisation (Bradley 1999).

In a stochastic context, a last-to-finish synchronisation can be represented by the maximum of two random sojourn times. Although perfectly possible to define a maximum distribution for two synchronising generally distributed actions (representing a last-to-finish style of synchronisation (Bradley 1999)), there is no closed form representation of the necessary Laplace transform for this (the product of cumulative distribution functions) so it would break our analytic tractability requirement to try to define it, by default. If we were going to resort to simulation, this would be a perfectly reasonable synchronisation model to choose.

As already discussed, semi-Markov operation in a model has quite specialised architectural applications (i.e. mutual exclusion, single processor execution). In semi-Markov PEPA, we allow a user-definable active synchronisation style so as not to restrict modelling expressiveness.

So for $P \xrightarrow{(a,\omega:L(s))} P'$ and $Q \xrightarrow{(a,\psi:N(s))} Q'$ then we take:

$$P \underset{\{a\}}{\bowtie} Q \xrightarrow{(a,\omega':R(s))} P' \underset{\{a\}}{\bowtie} Q'$$

where $\omega' = f(\omega, \psi)$ and $R(s) = g(\omega, \psi, L(s), N(s))$ are user-definable. We think of the combined a-actions as a single event in the cooperation, which derives its likelihood of occurrence and duration from the weights and distributions attributed to its constituents actions.

As an example, we might take $f(\omega, \psi) = \min(\omega, \psi)$ and $g(\omega, \psi, L(s), N(s)) = \frac{\omega}{\psi + \omega} L(s) + \frac{\psi}{\psi + \omega} N(s)$. Here such the weight of the overall event is driven by the weight of the action that is less likely to happen, hence the minimum function for ω'. The combined sojourn time, reflected in the transform, $R(s)$ would represent a probabilistic selection of either of the component sojourn times driven by the relative component weights. This would be one possible synchronisation definition is SM-PEPA.

4 SM-PEPA ANALYSIS

Figure 1 describes a semi-Markov PEPA example of a simple voting model with three voters and two poll-takers. Note that the semi-Markov aspect of the system is derived from the gamma distribution attached to the mutually exclusive recover_all action. This is an example of the second type of semi-Markov system as defined in Section 2.2.

The voter model is expanded to its global state space using the expansion rules of Figures 4 and 5. This is translated into an underlying semi-Markov chain and from there passage time analysis can be performed. The expansion and analysis process is done mechanically by DNAmaca after ipc (the Imperial PEPA Compiler) (Bradley, Dingle, Gilmore & Knottenbelt 2003) has compiled the semi-Markov PEPA description into a semi-Markov stochastic Petri net (the input language for DNAmaca).

Figures 3 and 2 show two passage results from DNAmaca for the voter model. They show the passage time from the first observed vote action to the first full system recover, or recover_all action.

From Figure 2, we can get specific quantile information (useful for probabilistic quality-of-service guarantees) e.g. the probability that the passage from the first vote action to the first full system recover has taken place by time, $t = 27.5$, is 60.8%.

5 CONCLUSION

We have presented a semi-Markov version of PEPA, so that greater distribution expressiveness can be used within the PEPA environment. This will allow PEPA users to make use of recent advances in stochastic tool support, in particular provided by semi-Markov DNAmaca.

The addition of priority levels each of which can have concurrent Markovian actions or probabilistic but generally distributed semi-Markov actions is an important addition to the modelling arsenal. We envisage the use of priority labels rather than numeric priorities in practice, with a partial order expressed by the user. The translation to numeric labels would then occur at the tool level to allow the semantics expressed here to be used.

Acknowledgements

I would like to thank Nick Dingle for help in generating the results from DNAmaca and also Tony Field, Stephen Gilmore, Jane Hillston, Leïla Kloul, Nigel Thomas and the anonymous referees for very useful comments and suggestions made on earlier drafts. Jeremy Bradley is supported in part by the Nuffield Foundation under grant NAL/00805/G.

REFERENCES

Bowman, H., Bryans, J. W. & Derrick, J. (2001), 'Analysis of a multimedia stream using stochastic process algebras', *The Computer Journal* **44**(4), 230–245.

Bradley, J. T. (1999), Towards Reliable Modelling with Stochastic Process Algebras, PhD thesis, Department of Computer Science, University of Bristol, Bristol BS8 1UB, UK.

Bradley, J. T. (2003), Semi-Markov PEPA: a contradiction in terms?, *in* S. T. Gilmore, ed., 'PASTA'03, Proceedings of 2nd Workshop on Process Algebras and Stochastically Timed Activities', LFCS, Edinburgh, pp. 1–6.

Bradley, J. T. (2004), Semi-Markov PEPA: Compositional modelling and analysis with generally distributed actions, *in* I. Awan, ed., 'UKPEW'04, Proceedings of 20th Annual UK Performance Engineering Workshop', University of Bradford, pp. 266–275.

Bradley, J. T. & Davies, N. J. (1999), Reliable performance modelling with approximate synchronisations, *in* Hillston & Silva (1999), pp. 99–118.

Bradley, J. T., Dingle, N. J., Gilmore, S. T. & Knottenbelt, W. J. (2003), Derivation of passage-time densities in PEPA models using ipc: the Imperial PEPA Compiler, *in* G. Kotsis, ed., 'MASCOTS'03, Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems', IEEE Computer Society Press, University of Central Florida, pp. 344–351.

Bradley, J. T., Dingle, N. J., Knottenbelt, W. J. & Harrison, P. G. (2003), Performance queries on semi-Markov stochastic Petri nets with an extended Continuous Stochastic Logic, *in* G. Ciardo & W. Sanders, eds, 'PNPM'03, Proceedings of Petri Nets and Performance Models', IEEE Computer Society, University of Illinois at Urbana-Champaign, pp. 62–71.

Bravetti, M., Bernardo, M. & Gorrieri, R. (1998), Towards performance evaluation with general distributions in process algebras, *in* D. Sangiorgi & R. de Simone, eds, 'CONCUR'98, Proceedings of the 9th International Conference on Concurrency Theory', Vol. 1466 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 405–422.

Bravetti, M. & Gorrieri, R. (1999), Interactive Generalized Semi-Markov Processes, *in* Hillston & Silva (1999), pp. 83–98.

Clark, G. (1994), Techniques for the Construction and Analysis of Algebraic Performance Models, PhD thesis, Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, UK. CST–107–94.

D'Argenio, P. R. (1999), Algebras and Automata for Timed and Stochastic Systems, PhD thesis, Department of Computer Science, University of Twente.

D'Argenio, P. R., Katoen, J.-P. & Brinksma, E. (1998), A compositional approach to Generalised Semi-Markov Processes, *in* 'WODES'98, Proceedings of the 4th International Workshop on Discrete Event Systems', IEE, Caligari, pp. 391–387.

Dugan, J. B., Trivedi, K. S., Geist, R. M. & Nicola, V. F. (1984), Extended stochastic Petri nets: Applications and analysis, *in* E. Gelenbe, ed., 'Performance'84, Proceedings of 10th Intl. Symp. on Models of Computer System Performance', Elsevier, pp. 507–519.

El-Rayes, A., Kwiatkowska, M. & Norman, G. (1999), Solving infinite stochastic process algebra models through matrix-geometric methods, *in* Hillston & Silva (1999), pp. 41–62.

Harrison, P. G. & Strulo, B. (1995), 'Process algebra for discrete event simulation', *Qualitative Methods in Parallel Systems*.

Hillston, J. (1994), The nature of synchronisation, *in* U. Herzog & M. Rettelbach, eds, 'Process Algebra and Performance Modelling Workshop', Arbeitsberichte des IMMD, Universtät Erlangen-Nürnberg, pp. 49–68.

Hillston, J. (1996), *A Compositional Approach to Performance Modelling*, Vol. 12 of *Distinguished Dissertations in Computer Science*, Cambridge University Press. ISBN 0 521 57189 8.

Hillston, J. & Silva, M., eds (1999), *PAPM'99, Proceedings of the 7th International Workshop on Process Algebra and Performance Modelling*, Centro Politécnico Superior de la Universidad de Zaragoza, Prensas Universitarias de Zaragoza. ISBN 84 7733 513 3.

PEPA Resource Site (n.d.), `http://www.dcs.ed.ac.uk/pepa/`.

Pyke, R. (1961), 'Markov renewal processes: Definitions and preliminary properties', *Annals of Mathematical Statistics* **32**(4), 1231–1242.

Biography

Jeremy Bradley is a lecturer in the Department of Computing at Imperial College London. His research has produced novel algorithms for transient and response time analysis of semi-Markov processes. His other activities include looking at techniques to represent and reason with complex performance measures in high-level formalisms such as stochastic Petri nets and stochastic process algebras. He wrote and maintains the Imperial PEPA compiler, which analyses PEPA process algebra models and performance measures using the DNAmaca tool. He is a self-confessed Haskell junky.

A voter can cast a vote and then pause before becoming a voter again.

$$\mathrm{Voter} \stackrel{\mathit{def}}{=} (\mathrm{vote}, \lambda).(\mathrm{pause}, \mu).\mathrm{Voter}$$

A *poll-taker* machine can receive a vote or break; if it breaks it can recover or wait for a global recover_all[2].

$$\mathrm{Poll\_taker} \stackrel{\mathit{def}}{=} (\mathrm{vote}, \top).(\mathrm{register}, \gamma).\mathrm{Poll\_taker} + (\mathrm{break}, \nu).\mathrm{Poll\_taker\_broken}$$
$$\mathrm{Poll\_taker\_broken} \stackrel{\mathit{def}}{=} (\mathrm{recover}, \tau).\mathrm{Poll\_taker} + (\mathrm{recover\_all}^{[2]}, \top).\mathrm{Poll\_taker}$$

The Poll_taker_group is used to manage many parallel poll-takers. If they all break then the Poll_taker_group_2 can initiate a high priority semi-Markov recover_all[2] action that can repair all of the poll-taker units simultaneously—essentially a system reset.

$$\mathrm{Poll\_taker\_group\_0} \stackrel{\mathit{def}}{=} (\mathrm{break}, \top).\mathrm{Poll\_taker\_group\_1}$$
$$\mathrm{Poll\_taker\_group\_1} \stackrel{\mathit{def}}{=} (\mathrm{break}, \top).\mathrm{Poll\_taker\_group\_2} + (\mathrm{recover}, \top).\mathrm{Poll\_taker\_group\_0}$$
$$\mathrm{Poll\_taker\_group\_2} \stackrel{\mathit{def}}{=} (\mathrm{recover\_all}^{[2]}, \mathit{1} : \mathtt{gamma}(\delta, \mathtt{k}, \mathtt{s})).\mathrm{Poll\_taker\_group\_0}$$

The system definition combines 3 voters with 2 poling units.

$$\mathrm{System} \stackrel{\mathit{def}}{=} (\mathrm{Voter} \parallel \mathrm{Voter} \parallel \mathrm{Voter}) \underset{\{\mathrm{vote}\}}{\bowtie} ((\mathrm{Poll\_taker} \underset{L}{\bowtie} \mathrm{Poll\_taker}) \underset{L'}{\bowtie} \mathrm{Poll\_taker\_group\_0})$$

where $L = \{\mathrm{recover\_all}\}$, $L' = \{\mathrm{recover}, \mathrm{break}, \mathrm{recover\_all}\}$

Fig. 1. Semi-Markov PEPA description of a simple voting model

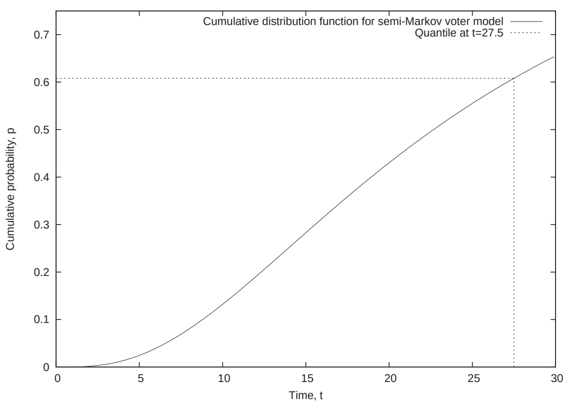

Fig. 2. Passage time cumulative density function and quantile for the time taken from the first cast vote to the first full failure recovery.

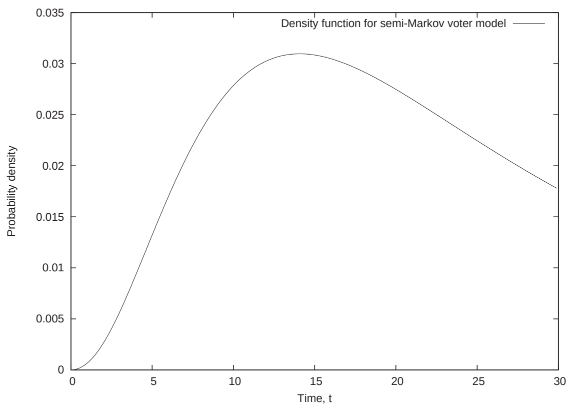

Fig. 3. Passage time density function for the time taken from the first cast vote to the first full failure recovery.

Prefix

$$\frac{}{(a^{[n]}, D).P \xrightarrow{(a^{[n]},D)} P}$$

Competitive Choice

$$\frac{P \xrightarrow{(a^{[n]},\lambda)} P'}{P + Q \xrightarrow{(a^{[n]},\lambda)} P'} \text{ if } Q \stackrel{(b^{[m]},D)}{\not\longrightarrow} \text{ where } m > n$$

$$\frac{Q \xrightarrow{(a^{[n]},\mu)} Q'}{P + Q \xrightarrow{(a^{[n]},\mu)} Q'} \text{ if } P \stackrel{(b^{[m]},D)}{\not\longrightarrow} \text{ where } m > n$$

Cooperation

$$\frac{P \xrightarrow{(a^{[n]},\lambda)} P'}{P \underset{S}{\bowtie} Q \xrightarrow{(a^{[n]},\lambda)} P' \underset{S}{\bowtie} Q} \text{ if } a \notin S \text{ and } Q \stackrel{(b^{[m]},D)}{\not\longrightarrow} \text{ where } m > n$$

$$\frac{Q \xrightarrow{(a^{[n]},\mu)} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a^{[n]},\mu)} P \underset{S}{\bowtie} Q'} \text{ if } a \notin S \text{ and } P \stackrel{(b^{[m]},D)}{\not\longrightarrow} \text{ where } m > n$$

$$\frac{P \xrightarrow{(a^{[n]},\lambda)} P' \quad Q \xrightarrow{(a^{[n]},\mu)} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a^{[n]},R)} P' \underset{S}{\bowtie} Q'} \text{ if } a \in S$$

$$\text{where } R = \frac{\lambda}{r_a^n(P)} \frac{\mu}{r_a^n(Q)} \min(r_a^n(P), r_a^n(Q))$$

Hiding

$$\frac{P \xrightarrow{(a,D)} P'}{P \backslash L \xrightarrow{(a,D)} P' \backslash L} \text{ if } a \notin L \qquad \frac{P \xrightarrow{(a,D)} P'}{P \backslash L \xrightarrow{(\tau,D)} P' \backslash L} \text{ if } a \in L$$

Constant

$$\frac{P \xrightarrow{(a,D)} P'}{A \xrightarrow{(a,D)} P'} \; A \stackrel{\text{def}}{=} P$$

Fig. 4. Operational semantics for SM-PEPA: normal Markovian semantics are preserved if only Markovian actions are enabled.

Competitive Choice

$$\frac{P \xrightarrow{(a^{[n]},\omega:L(s))} P'}{P + Q \xrightarrow{(a^{[n]},\omega:L(s))} P'} \quad \text{if } Q \stackrel{(b^{[m]},D)}{\not\longrightarrow} \text{ where } m > n$$

$$\frac{Q \xrightarrow{(a^{[n]},\omega:L(s))} Q'}{P + Q \xrightarrow{(a^{[n]},\omega:L(s))} Q'} \quad \text{if } P \stackrel{(b^{[m]},D)}{\not\longrightarrow} \text{ where } m > n$$

Cooperation

$$\frac{P \xrightarrow{(a^{[n]},\omega:L(s))} P'}{P \underset{S}{\bowtie} Q \xrightarrow{(a^{[n]},\omega:L(s))} P' \underset{S}{\bowtie} Q} \quad \text{if } a \notin S \text{ and } Q \stackrel{(b^{[m]},D)}{\not\longrightarrow} \text{ where } m > n$$

$$\frac{Q \xrightarrow{(a^{[n]},\omega:L(s))} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a^{[n]},\omega:L(s))} P \underset{S}{\bowtie} Q'} \quad \text{if } a \notin S \text{ and } P \stackrel{(b^{[m]},D)}{\not\longrightarrow} \text{ where } m > n$$

$$\frac{P \xrightarrow{(a^{[n]},\omega:L(s))} P' \quad Q \xrightarrow{(a^{[n]},\top)} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a^{[n]},\omega:L(s))} P' \underset{S}{\bowtie} Q'} \quad \text{if } a \in S$$

$$\frac{P \xrightarrow{(a^{[n]},\top)} P' \quad Q \xrightarrow{(a^{[n]},\omega:L(s))} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a^{[n]},\omega:L(s))} P' \underset{S}{\bowtie} Q'} \quad \text{if } a \in S$$

$$\frac{P \xrightarrow{(a^{[n]},\omega:L(s))} P' \quad Q \xrightarrow{(a^{[n]},\psi:N(s))} Q'}{P \underset{S}{\bowtie} Q \xrightarrow{(a^{[n]},\omega':R(s))} P' \underset{S}{\bowtie} Q'} \quad \text{if } a \in S$$

where $\omega' = f(\omega, \psi)$ and $R(s) = g(\omega, \psi, L(s), N(s))$ are user-definable synchronisation functions

Fig. 5. Operational semantics for SM-PEPA: semi-Markov extensions