Constructing a Partial Order for Performance Measures*

Jeremy Bradley† Nigel Thomas‡

Abstract

This paper describes a general performance-related partial order over stochastic systems. A partial order enables a designer to optimise a model for specific performance features, such as usage, throughput and bandwidth. The partial order presented here enables comparisons to be made of representations of systems at different levels of abstraction. This could potentially generate reward-based equivalences that would significantly reduce the computed state-space when a system is analysed for a particular property.

1 Introduction

One of the common ways in which stochastic systems are analysed is by examining their steady-state behaviour: that is, the proportion of time spent in each of the states averaged over the long term. This provides a discrete distribution to which reward vectors can be applied and an expected reward calculated. What this reward measures depends on how it was constructed relative to the states of the system it was rewarding. This makes the construction of an appropriate reward vector (so-called *reward specification*) for a particular system quality (e.g. bandwidth, throughput, usage) a task which has been of interest in stochastic process algebras [1, 2, 3, 4].

In particular, it would be especially nice to be able to compare two systems with respective reward structures and say that one performs *better* than the other. This, in informal terms, is what we mean by a partial order when applied to

*EPSRC funded QWiD project GR/N01491.

†Email: `Jeremy.Bradley@durham.ac.uk`

‡Department of Computer Science, The University of Durham, Science Laboratories, South Road, Durham, DH1 3LE. Email: `Nigel.Thomas@durham.ac.uk`

stochastic systems. At first sight, this appears to be a trivial and uninformative exercise. It might be assumed that systems need to be of the same size and the reward structures need to be identical, before a valid comparison can be made. Then, it is simply a matter of calculating the reward for each and comparing the two.

This paper investigates ways of generalising both the systems being compared and the reward vectors associated with them in such a way that a meaningful comparison can still be made. In doing so, we also see a way of constructing reward-based equivalences which might be used to refine a model while still maintaining a desired property for a system. This ability, to be able to talk about the local structure of a system while being able to keep an eye on a global property, would be a very useful departure from current locally-defined equivalences and would go some way to achieving one of the future goals set out in [5].

In the meantime, the development of a general performance partial order was one of the useful research avenues identified by Hillston [6]. At a tool level this could be used to aid the designer in understanding how alterations to a system design would adversely or beneficially affect a particular property.

2 Definitions

Traditional steady-state analysis relies on a stochastic system's being ergodic and possessing a steady-state distribution, p_i. A reward vector is a state weighting vector, r_i, which represents a particular property that a designer wishes to measure. A reward R is constructed from:

$$R = \sum_i p_i r_i \tag{1}$$

3 Partial Orders and Reward Vectors

Before we launch into an algebraic description of the partial order, we will attempt to outline what we are trying to achieve.

We are trying to create the standard binary partial order relationship ($\leq$) between pairs of systems operated on by particular reward vectors. In order for this to be as useful a relation as possible we want to generalise the form of the reward vector that can legitimately be used in a comparison.

Clearly arbitrary reward vectors describe different properties of a system and just as it would make no sense to compare average queue length with average throughput, say, it therefore makes no sense to compare systems being rewarded by completely unrelated reward vectors. On the other hand, we can do better than insist that systems can only be compared in the partial order only if they are being rewarded with identical vectors.

One of our aims is to allow comparisons between systems with reward structures where one is an aggregate representation of the other. This allows systems which represent different abstractions of the same thing to be compared, i.e. a low-level and a high level description of a system would fall into this category.

We can further generalise the idea to allow comparison between systems which, at different stages in their state-space, represent high and low level design of the same underlying model. This is achieved by allowing comparison between reward vectors which can be individually aggregated to form a third common aggregate reward structure.

4 Definition of the Partial Order

Formally, we require a relation, $\leq$, between the tuples of the form (S, R_S) where S is the set of states of a system with a steady-state distribution and R_S is the reward vector over that system representing a particular property. If the tuples A, B are related by $A \leq B$ then it implies that the overall reward accrued by tuple A is less than or equal to that of tuple B.

We define f_S to be a reward function on a system, S, such that:

$$f_S :: \mathsf{STATE} \longrightarrow \mathbb{R}^+ \cup \{0\} \tag{2}$$

$$R_S = \{f_S(s) : s \in S\} \tag{3}$$

We also need an equivalent steady-state function on a system, S:

$$\pi_S :: \mathsf{STATE} \longrightarrow [0, 1] \tag{4}$$

$$\sum_{\forall s \in S} \pi_S(s) = 1 \tag{5}$$

Now we can say $(S_1, R_{S_1}) \leq (S_2, R_{S_2})$ is valid if $R_{S_1} \sim R_{S_2}$, where $(\sim)$ sets out exactly how the reward vectors have to be related in order for the partial order to be meaningful.

We define our primitive aggregation on a reward structure (this is explained in more detail in section 4.1):

Definition 4.1 $R_S \longrightarrow R_{S'}$ *if* $R_{S'} = R_S$ *or* $\exists\ r_1 = f_S(s_1), r_2 = f_S(s_2) \in R_S$ *such that:*

$$R_{S'} = (R_S \setminus \{r_1, r_2\}) \cup \{\sigma_S(s_1, s_2)\} \tag{6}$$

where

$$\sigma_S(a, b) = \frac{\pi_S(a) f_S(a) + \pi_S(b) f_S(b)}{\pi_S(a) + \pi_S(b)} \tag{7}$$

This allows us to define a relation between two reward vectors that have undergone a sequence of zero or more such primitive aggregations:

Definition 4.2 $R \stackrel{*}{\longrightarrow} R'$ *if* $\exists\ n > 0$ *and a sequence of vectors* $R^{(1)}$, $R^{(2)}, \ldots,$ $R^{(n)}$ *such that* $R' = R^{(n)}$ *and:*

$$R \longrightarrow R^{(1)} \longrightarrow R^{(2)} \longrightarrow \ldots \longrightarrow R^{(n)} \tag{8}$$

We can now define the original relation, $(\sim)$, which sets out the third common aggregate reward, as follows:

Definition 4.3 $R_{S_1} \sim R_{S_2}$ *if and only if* $\exists\ \rho$ *such that* $R_{S_1} \stackrel{*}{\longrightarrow} \rho$ *and* $R_{S_2} \stackrel{*}{\longrightarrow} \rho$

Finally, given $R_{S_1} \sim R_{S_2}$ is true we can state the expected partial order condition:

Definition 4.4 $(S_1, R_{S_1}) \leq (S_2, R_{S_2})$ *iff:*

$$\sum_{\forall s \in S} \pi_{S_1}(s) R_{S_1}(s) \leq \sum_{\forall s \in S} \pi_{S_2}(s) f_{S_2}(s) \tag{9}$$

4.1 Explanation

What has been achieved? We have stipulated a necessary condition on the form of a pair of reward structures to allow them and their systems to be compared in a partial order relation. We particularly need to explain the form of $\sigma_S(\cdot, \cdot)$.

This form of state aggregation is based upon the premise that we can arbitrarily add any two entries in the steady-state distribution of a system; specifically, in applying the following transformation:

$$g(\pi_1, \pi_2, \pi_3, \ldots) = (\pi_1 + \pi_2, \pi_3, \ldots) \tag{10}$$

All we have done is amalgamated the long term probability that the system will end up in state 1 or 2 of the system as $\pi_1 + \pi_2$. This seems like an obvious way of expressing a higher-level abstraction of a system. If the above system had had an associated reward vector prior to the transformation: $(r_1, r_2, \ldots)$ then in order for the same overall reward to be accumulated by the system, it is necessary that:

$$\pi_1 r_1 + \pi_2 r_2 = r'(\pi_1 + \pi_2) \tag{11}$$

where r' represents the new reward figure for the transformed system. This, then, is the reason for the form of $\sigma_S(\cdot, \cdot)$ given in equation (7).

What the relation ($\sim$) is saying is that if we can reduce two systems' reward structures iteratively in this way until they have a *common* reward structure then the two systems are comparable, as they were originally being rewarded for the same property, only at different levels of abstraction.

4.2 Example

We present an example of what has been described in the previous section. We have two systems S and T which have the following steady-state distributions:

$$\pi_S = \left(\frac{3}{20}, \frac{1}{5}, \frac{1}{2}, \frac{1}{20}, \frac{1}{40}, \frac{1}{20}, \frac{1}{40}\right) \tag{12}$$

$$\pi_T = \left(\frac{3}{20}, \frac{7}{10}, \frac{1}{10}, \frac{1}{120}, \frac{1}{60}, \frac{1}{40}\right) \tag{13}$$

They are rewarded by different reward structures, R_S and R_T:

$$R_S = (8, 6, 3, 0, 1, 1, 2) \tag{14}$$

$$R_T = \left(8, \frac{27}{7}, 0, 2, \frac{1}{2}, \frac{4}{3}\right) \tag{15}$$

The reduction using the transformation described above is shown in Figure 1 and shows that $R_S \sim R_T$ and thus a partial ordering is valid. It only remains to note that $\sum_{\forall s \in S} f_S(s)\pi_S(s) = 4\frac{1}{40}$ and $\sum_{\forall t \in T} f_T(t)\pi_T(t) = 3\frac{14}{15}$. Thus we can say that $(S, R_S) \leq (T, R_T)$ and for the particular property represented by R_S and R_T, S performs better than T.

$$
\begin{array}{llcll}
\pi_S & \left(\frac{3}{20}, \frac{1}{5}, \frac{1}{2}, \frac{1}{20}, \frac{1}{40}, \frac{1}{20}, \frac{1}{40}\right) & & & \\
R_S & \left(8, 6, 3, 0, 1, \underbrace{1, 2}\right) & & & \\
 & \quad\downarrow & & & \\
\pi_{S'} & \left(\frac{3}{20}, \frac{1}{5}, \frac{1}{2}, \frac{1}{20}, \frac{1}{40}, \frac{3}{40}\right) & & \left(\frac{3}{20}, \frac{7}{10}, \frac{1}{10}, \frac{1}{120}, \frac{1}{60}, \frac{1}{40}\right) & \pi_T \\
R_{S'} & \left(8, \underbrace{6, 3}, 0, 1, \frac{4}{3}\right) & & \left(8, \frac{27}{7}, 0, \underbrace{2, \frac{1}{2}}, \frac{4}{3}\right) & R_T \\
 & \quad\downarrow & & \quad\downarrow & \\
\pi_{S''} & \left(\frac{3}{20}, \frac{7}{10}, \frac{1}{20}, \frac{1}{40}, \frac{3}{40}\right) & & \left(\frac{3}{20}, \frac{7}{10}, \frac{1}{10}, \frac{1}{40}, \frac{1}{40}\right) & \pi_{T'} \\
R_{S''} & \left(8, \frac{27}{7}, 0, 1, \frac{4}{3}\right) & = & \left(8, \frac{27}{7}, 0, 1, \frac{4}{3}\right) & R_{T'}
\end{array}
$$

Figure 1: The reduction of two reward vectors to a common vector.

4.3 System Comparison

To demonstrate how this might be used at a system level, we present a model of a web server (Figure 2) in PEPA [7]. The global state space of this system is shown in Figure 3. Using state aggregation techniques developed in [5], for instance, we could aggregate the set of central states outlined in Figure 3 to generate a higher-level model of the same system, presented in Figure 4.

$$
\begin{array}{rcl}
\mathit{user} & \stackrel{\mathrm{def}}{=} & (\mathrm{req}, \iota_1).\mathit{user}_1 \\
\mathit{user}_1 & \stackrel{\mathrm{def}}{=} & (\mathrm{think}, \iota_2).(\mathrm{html}, \top).\mathit{user} + (\mathrm{html}, \top).(\mathrm{think}, \iota_2).\mathit{user} \\
\mathit{browser} & \stackrel{\mathrm{def}}{=} & (\mathrm{req}, \top).(\mathrm{GET}, \lambda_1).(\mathrm{html}, \top).\mathit{browser} \\
\mathit{httpd} & \stackrel{\mathrm{def}}{=} & (\mathrm{GET}, \top).(\mathrm{fetch}, \mu_1).(\mathrm{html}, \mu_2).\mathit{httpd} \\
\mathit{system} & \stackrel{\mathrm{def}}{=} & (\mathit{browser} \underset{\{\mathrm{GET},\mathrm{html}\}}{\bowtie} \mathit{httpd}) \underset{\{\mathrm{req},\mathrm{html}\}}{\bowtie} \mathit{user}
\end{array}
$$

Figure 2: A PEPA model of a user, browser and web server.

For the sake of argument, let us assume that we can set the parameters in

Figure 3 such that X has a uniform steady-state distribution:

$$\pi_X = \left(\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}\right) \tag{16}$$

By steady-state aggregation we can infer that the system, Y, of Figure 4 has steady-state distribution:

$$\pi_Y = \left(\frac{1}{8}, \frac{3}{4}, \frac{1}{8}\right) \tag{17}$$

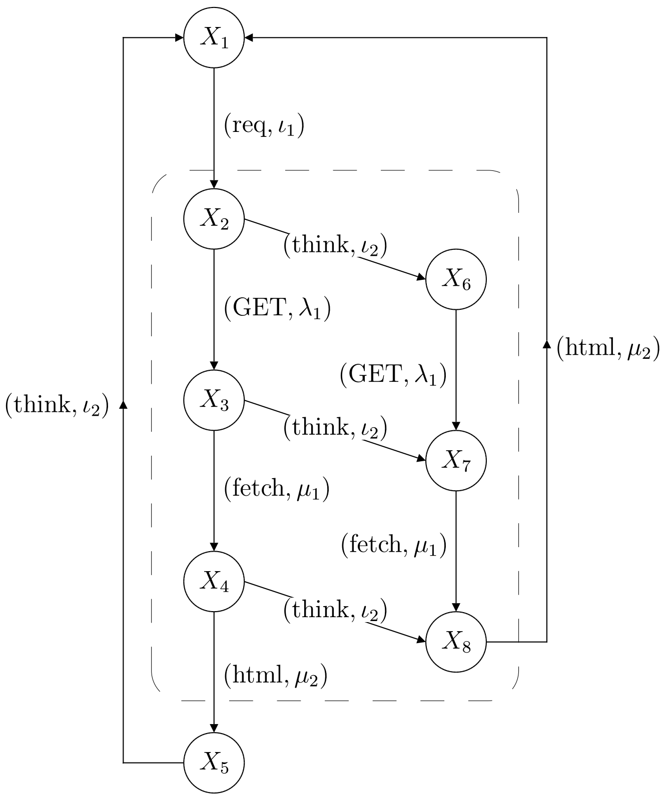

Figure 3: The global state-space of the PEPA web server model.

We can now construct the reward structures R_X and R_Y which both measure the percentage of time the system takes to return the token `html`:

$$R_X = (0, 0, 0, 0, 1, 0, 0, 1) \tag{18}$$

Assuming that state X_1 has been mapped onto Y_1 and state X_5 onto Y_3, then using the reduction of Figure 1 we get:

$$R_Y = \left(0, \frac{1}{6}, r_3\right) \tag{19}$$

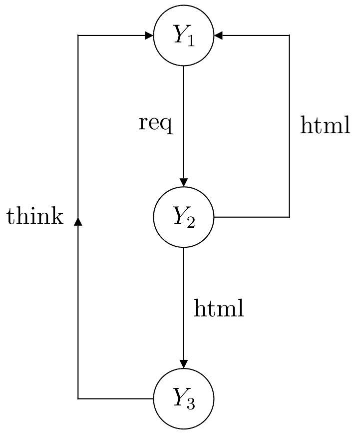

Figure 4: The aggregated global state-space of the web server model.

if X and Y are to be comparable, i.e. $R_X \sim R_Y$. If $r_3 = 1$ then the rewards for X and Y are the same.

What this highlights is the relative nature of reward structures. In selecting a reward for an individual system, beyond being relatively consistent between its own states, it can have any arbitrary scaling factor applied to it. What the ($\sim$) relation attempts to do is maintain relative consistency of reward setting across different systems.

Another feature of this example is that we have aggregated the global state space directly. This was really for simplicity and ease of demonstration. We are really much more interested in performing component-level aggregation on the process algebra itself and then determining what might need to be altered in a reward specification to be consistent with the reward structure of the unaggregated system. This is clearly a much more desirable approach from a system modelling point of view.

5 Conclusions

We have outlined, in this paper, a partial order for relating pairs of steady-state distributions and reward structures. Essentially we have shown that, under certain restrictions, systems can be compared for the same rewarded property even if they have wildly different representations. This relation is only, as yet, defined on the global state-space which has the benefit that it can be applied to any stochastic system (SPNs, SPAs or SANs for instance) and the drawback that the global state-space has to be explicitly generated. We intend to concentrate on SPAs in future work to see what effect aggregating components has on global reward structures and properties.

It is a simple enough procedure to turn a partial order into an equivalence for a system (let $A = B$ iff $A \leq B$ and $B \leq A$) and given this, we would like to consider how to restrict the generation of the state-space so that only sufficient is created to generate a given performance measure. This may best be achieved at the component description level of a stochastic process algebra. If we can find an equivalence which is related to a global reward measure but expressible at a component or transition level then the potential for reducing the global state-space is huge. Traditional Markovian equivalences tend to impose a rather strict notion upon a model since they are guaranteed to produce behaviourally identical systems which will have identical properties under any reward structure. Unless this is actually required in a model—maybe only one property is of interest—it would seem sensible to have a much more flexible equivalence which would only preserve the model's performance under a given reward structure.

This particular relation only deals with the situation where there is an underlying reward structure operating on different abstractions of a system. We have not dealt with the question of whether a property on a system might be represented equally well by a different reward vector, amongst other things. There is still further work to be done to find a necessary *and sufficient* relationship between reward vectors.

References

[1] G. Clark, "Formalising the specification of rewards with PEPA," in *Process Algebra and Performance Modelling Workshop* (M. Ribaudo, ed.), pp. 139–160, CLUT, Torino, July 1996.

[2] G. Clark, S. Gilmore, J. Hillston, and N. Thomas, "Experiences with the PEPA performance modelling tools," in *UKPEW'98, Proceedings of the 14th UK Performance Engineering Workshop*, Edinburgh, July 1998.

[3] S. Gilmore and J. Hillston, "Feature interaction in PEPA," in Priami [8], pp. 17–26.

[4] S. Gilmore and J. Hillston, "A feature construct for PEPA," in *UKPEW'99, Proceedings of the 15th Annual UK Performance Engineering Workshop* (J. T. Bradley and N. J. Davies, eds.), CSTR Technical Report, pp. 225–

236, Department of Computer Science, University of Bristol, Research Press, Bristol, July 1999.

[5] J. T. Bradley, *Towards Reliable Modelling with Stochastic Process Algebras.* PhD thesis, Department of Computer Science, University of Bristol, Bristol BS8 1UB, UK, October 1999.

[6] J. Hillston, "Exploiting structure in solution: Decomposing composed models," in Priami [8], pp. 1–15.

[7] J. Hillston, *A Compositional Approach to Performance Modelling*, vol. 12 of *Distinguished Dissertations in Computer Science.* Cambridge University Press, 1996. ISBN 0 521 57189 8.

[8] C. Priami, ed., *PAPM'98, Proceedings of the 6th International Workshop on Process Algebra and Performance Modelling*, Università Degli Studi di Verona, Nice, September 1998.