The Nature of Synchronisation[*]

Jane Hillston

August 1994

Abstract

In each of the current stochastic process algebras all non-competitive interactions between components or agents are modelled using a single combinator, variously called the *parallel*, *synchronisation* or *cooperation* operator. This paper aims to compare the definitions of this combinator which have been used; in particular, looking at the different ways in which rates are associated with the actions which result from such interactions. The implications of the chosen definitions, from a modelling point of view, will be described.

When we consider concrete systems rather than abstract representations many different types of interactions between systems are exhibited. Some of these possible interactions are presented in the latter half of the paper and we analyse the extent to which these can be captured using the combinators available in the SPA languages. To conclude some observations about current modelling practice are made together with suggestions of potential extensions to the set of combinators.

1 Introduction

In recent years there has been growing interest in the use of a process algebra-based modelling paradigm for the performance evaluation of computer and communication systems. This interest has resulted in the development of extensions of pure process algebras in which time and probabilistic elements are introduced, *stochastic process algebras* (SPA). As in a pure process algebra, a system is modelled as an interaction of *agents*; the behaviour of each agent is defined by the *actions* it can perform or as a composition of smaller agents. However, actions are no longer assumed to be instantaneous. Each has a duration characterised by an exponentially distributed random variable.

There are several motivations for the use of a process algebra based paradigm for performance modelling, but perhaps the most important one is *compositionality*. A compositional approach offers the potential for complex systems to be modelled systematically. Separate aspects or components of a system may be considered in detail individually, but subsequently in a more abstract form as the interactions between them are developed. Moreover, the structure inherent in the model may often be exploited for model manipulation and analysis, reducing a complex task to a series of simpler ones.

However, compositionality is not without disadvantages. In particular, stochastic process algebras require the performance modeller to consider, for the first time, how to represent timed interactions between systems. In most existing performance modelling paradigms models are constructed as a single entity. This means that if two components within a model interact, the interaction is modelled implicitly, by a single action. In contrast, when a compositional approach is adopted, each component is modelled

[*]This paper appears in the Proceedings of the 2nd Workshop on Process Algebra and Performance Modelling, U. Herzog and M. Rettelbach (Eds), University of Erlangen, held in July 1994

separately; each submodel includes the potential to take part in the action which forms the basis of the interaction. The interaction between the components is modelled as an interaction of the submodels, the new action, or *interaction*, being derived from the individual actions of the two components.

There are some performance modelling paradigms in which models are constructed as an interaction of smaller parts, for example HIT [1] and stochastic automata networks (SAN) [2]. However, these methodologies restrict interactions so that only one participant defines the behaviour of the interaction. In HIT all interactions are limited to just two agents, who exhibit a *service* protocol: one agent offers a facility, the other agent uses it. Thus one of them controls the interaction. In SAN more general forms of interaction are allowed but there is still a restriction that only one of the participating actions carries timing information and this defines the interaction. It is possible that such limited forms of interaction are sufficient for modelling computer and communication systems, the usual application for performance modelling. This issue will be discussed in more detail in Section 5. However, for the remainder of this paper we assume that it is desirable to allow a more general form of interaction between components.

Compositionality does not lead to such problems in pure process algebras such as CCS [3] and CSP [4]. In these formalisms all actions are assumed to be instantaneous, thus all interactions are also instantaneous. Therefore, non-competitive interaction between agents is represented by the superposition of two (or possibly more, in the case of CSP) actions of the same type to form a single τ action (CCS) or a single action of the same type (CSP).

In probabilistic extensions of process algebra, such as PCCS [5], actions are still assumed to be instantaneous so interactions are derived as above. Similarly, in the timed extensions of CCS and CSP, TCCS [6] and Timed CSP [7], instantaneous actions are superposed to form instantaneous interactions, since timing is introduced via delays which are interleaved with actions. In Timed ACP [8] a (deterministic) time is associated with each action but this is regarded as an absolute or relative time stamp and actions are still assumed to be instantaneous. In this case an interaction is only seen to occur if the participating actions have the same time stamp.

Thus, it seems that stochastic process algebras cannot draw on established results to characterise the meaning of synchronisation between timed actions. It is then, perhaps, not surprising that the current SPA languages have all adopted slightly different solutions to this problem. Indeed the major differences between the languages are found in their different definitions for timed, non-competitive interaction between agents.

2 Synchronisation in Stochastic Process Algebras

All current stochastic process algebras have settled on the same core set of combinators: *prefix, choice, hiding* and *parallel composition* based on the CSP concurrency operator. This latter combinator is variously termed the *parallel, synchronisation* or *cooperation* operator in papers about the SPAs, but will be referred to as the *parallel* operator for the remainder of this paper. Interaction between agents is captured by the choice and parallel operators, representing competitive and non-competitive interactions respectively. It is assumed that in the course of a competitive interaction only one agent will complete some work and experience a state change because the agents are in competition for some implicit resource. In contrast, in a non-competitive interaction it is assumed, in general, that both participants will complete some work and progress, as each has its own implicit resource. For the purposes of this paper we define *synchronisation* as an action resulting from non-competitive interaction between agents which results in a state change in both participants.

The stochastic process algebras which will be discussed are listed below. All the languages are centred on actions which are assumed to have an exponentially distributed duration.

TIPP - **Ti**med **P**rocesses and **P**erformance Evaluation

TIPP was developed at the University of Erlangen by the group of Prof. Herzog. It evolved from an earlier language EXL, which was the first process algebra to be used for performance modelling. Actions are specified as a (type, rate) pair, where the rate is the parameter of the associated exponential distribution. For further details see [9, 10, 11, 12].

PEPA - **P**erformance **E**valuation **P**rocess **A**lgebra

PEPA was developed at the University of Edinburgh by Jane Hillston. As in TIPP actions are represented as (type, rate) pairs. PEPA is the smallest language in terms of combinators. Furthermore, a subset of the language terms which are guaranteed to result in ergodic Markov processes has been identified. These restrictions limit the way in which the combinators may be used within a model, but not which combinators are used. An overview of PEPA can be found in [13] and more detail is available in [14].

TIPP and PEPA are very similar. It is important to note that instances of the same action type may exhibit different activity rates in both languages. *Passive* actions, in which the rate is left unspecified, are included in both these formalisms.

MPA - **M**arkovian **P**rocess **A**lgebra

A stochastic process algebra called MPA has been developed at the University of Dortmund by Peter Buchholz [15, 16]. In the remainder of this paper this will be referred to as D-MPA. As in TIPP and PEPA actions are represented as pairs (α, r) where α represents the action type. However, it is now assumed that there is a fixed rate μ_α for each action of type α and r denotes the number of concurrently active instances of the action. There are no passive actions in D-MPA.

MPA - **M**arkovian **P**rocess **A**lgebra

Another stochastic process algebra called MPA has been developed at the University of Bologna by Prof. Gorrieri and his colleagues [17, 18, 19]. In the remainder of this paper this will be referred to as B-MPA. Actions are again represented as (type, rate) pairs, but in addition instantaneous, or *immediate*, actions are included in the language, denoted by rate ∞. Moreover, passive actions, with rate 0, play a more prominent rôle in this language, as will become apparent when we discuss synchronisation. B-MPA has the richest set of combinators but the limitations placed on the use of the parallel operator restrict the expressibility of the language.

In all the SPA languages the CSP concurrency operator is used as the basis for synchronisation, rather than the CCS hand-shake communication implied by conjugate names. In the performance scenario the notion of autonomous agents is a natural one, and synchronisation is assumed to occur on a designated set of action types. This allows the possibility of more than two agents participating in an interaction instead of the strict pairing of actions imposed by conjugation. In the computer and communication systems such multiple interactions often need to be represented. This operator is denoted $\|_S$ in TIPP, D-MPA and B-MPA, and $\underset{S}{\bowtie}$ in PEPA, and the set of action types, S, is called the *synchronisation set* or *cooperation set* respectively. In fact this is a family of combinators $\|_S$, one for each possible set of types S, since $P\|_S Q$ may have distinct behaviour from $P\|_T Q$ if S and T differ. The behaviour of the operator is defined by rules of the same form in all the languages:

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E\|_S F \xrightarrow{(\alpha,r)} E'\|_S F}\ (\alpha \notin S) \qquad\qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E\|_S F \xrightarrow{(\alpha,r)} E\|_S F'}\ (\alpha \notin S)$$

$$\frac{E \xrightarrow{(\alpha,r_1)} E' \quad F \xrightarrow{(\alpha,r_2)} F'}{E\|_S F \xrightarrow{(\alpha,R)} E'\|_S F'}\ (\alpha \in S)\ \ SC$$

However, the definitions vary in terms of how they assign a rate to the interaction, (α, R), which results when the agents *synchronise*, or *cooperate*, to achieve an action type in the set S, and the side condition SC which might be imposed to restrict when such an interaction is considered possible.

Implicit within the form of the rules expressed above there is an assumption that the interaction, or *shared* action, will have an exponentially distributed duration, in the same way as the individual actions of the agents. This assumption is a pragmatic one, although we will see during the discussion of possible interactions between systems, that it is not necessarily a justified one.

As explained in the previous Section, we would like the duration of the interaction to be derived from the durations of the individual actions which are superposed to achieve it. In other words, R should be a function of r_1 and r_2, $R = r_1 \otimes r_2$. We can derive practical conditions which we would like to impose on the operator $\otimes$. If synchronisation is intended to be an interaction in which each agent takes an equal rôle, it is logical to assume that the combinator is symmetric, which implies that $\otimes$ is commutative. In order for relations defined over the algebra to be a congruence we further require that $\otimes$ is associative, and distributive over addition. If we consider these last two requirements from a performance modelling perspective rather than an algebraic one it is not immediately clear that they are necessary. However, congruence relations are necessary to exploit the compositional features of SPA models. Consequently, all the current languages have chosen definitions for $\otimes$ which satisfy these algebraic requirements.

Before we discuss the choice of $\otimes$ made by each of the SPAs, it is interesting to note that none of the languages uses the definition $R = r_1$ subject to the side condition $SC \equiv r_1 = r_2$. This would restrict synchronisation to occur between agents who start with the same view of the shared action. In B-MPA the form of interaction allowed is more restrictive than this. In contrast, TIPP and PEPA aim to offer more general forms of interaction between agents in which the individual actions which make up the interaction may have differing characteristics. In D-MPA, since all actions of a given type must have a fixed rate, it initially appears that the condition is met. However, since synchronisation is defined in terms of the number of instances of an action the rule does, in fact, allow more general interactions, as in the case of TIPP. We will return to a discussion of this special case in Section 3. Meanwhile in the following subsections we describe how $\otimes$ and SC are defined, and their implications, in each of the formalisms.

2.1 TIPP

$R = r_1 \times r_2 \qquad SC \equiv \emptyset$

The use of multiplication has a clear practical appeal: it is straightforward to calculate and satisfies all the algebraic requirements outlined above. Indeed, the primary justification given for this definition is the algebraic one.

In TIPP passive actions are represented with rate 1 so that they are algebraically neutral with respect to multiplication. When an agent is passive with respect to a given action type it may have several interpretations, but perhaps the most important is the idea of *service*. If the action represents a service which is provided by one agent and required by the other, it is natural that one of the participants in the interaction will dominate the behaviour. In TIPP it is assumed that the server will determine the rate of the interaction while the customer is passive.

A generalisation of this notion is obtained when some rates are assumed to represent a *scaling factor*, not a rate. In this case the individual action represents a workload requirement which is to be satisfied by interaction as in the case of a passive action outlined above. It is also assumed that there is some standard requirement for such service. The scaling factor is used to express the ratio of the standard requirement to the workload requirement of this agent. The mean duration of the service interaction is then scaled accordingly, when rate of the service action is multiplied by the scaling factor.

There are several problems with this approach, perhaps the most important of which is the lack of syntactic distinction that it makes between active elements of a model and workload. The representation of service interactions will be discussed in more detail in Section 5.

2.2 PEPA

$$R = \frac{r_1}{r_\alpha(E)}\frac{r_2}{r_\alpha(F)}\min(r_\alpha(E), r_\alpha(F)) \qquad SC \equiv \emptyset$$

The emphasis in PEPA is on the idea of *cooperation*—both participants must complete some work before the interaction can be achieved. The definition of $\otimes$ is chosen to reflect the idea that the rate at which such a joint action can occur will be limited by the rate of the slowest participant. $\min(\cdot,\cdot)$ is not in general distributive over addition. However, this problem is overcome when the rate of the interaction is expressed in terms of the total capacity of each of the participants to carry out an action of this type. This total capacity of an agent to carry out actions of a given type is called the *apparent rate of* α, $r_\alpha(E)$. To determine the apparent rate of the interaction the minimum of the apparent rates of the participants is used. However, this must then be renormalised over the instances of the action type which may actually be completed in the two agents. This is achieved by multiplying the apparent rate of the interaction by the conditional probabilities $r_1/r_\alpha(E)$ and $r_2/r_\alpha(F)$. It is assumed that these choices are made independently.

Clearly this is much more complicated to calculate than the product which is used in TIPP. Moreover this calculation is based on a more global "state" of the system, since the apparent rate of all participants needs to be calculated.

In PEPA passive actions are assumed to have *unspecified* rate, denoted $\top$ (called "top") which is the neutral element for $\min$. When more than one passive action competes to take part in an interaction weights must be assigned to ensure the correct probabilistic outcome.

2.3 D-MPA

Rate of interaction = μ_α No. of interactions = $R = r_1 \times r_2$ $SC \equiv \emptyset$

In D-MPA an action of type α must occur with a fixed rate μ_α so it follows that the interaction of two α actions results in an action α with rate μ_α. In this language,

$$(\alpha, r_1).E \xrightarrow{(\alpha,r_1)} E'$$

denotes that there are r_1 concurrently active instances of the action a, each of which, upon completion, results in the derivative E'. It is assumed that when an interaction occurs between E and F each α action of E has the possibility of interacting with each α action of F. Thus the total number of instances of α which result from the interaction is the product, $r_1 \times r_2$.

If $(\alpha, r_1).E \xrightarrow{(\alpha,r_1)} E'$, from the point of view of an external observer, this is indistinguishable from a single instance of an α action with rate $r_1 \times \mu_\alpha$. Thus the behaviour of synchronisation in D-MPA bears a close resemblance to the behaviour of synchronisation in TIPP. However, as will be discussed in Section 3, the two are not exactly the same.

There are no passive actions in D-MPA.

2.4 B-MPA

$R = \max(r_1, r_2) \qquad SC \equiv \min(r_1, r_2) = 0$

The side condition SC implies that at least one of the participating agents must be passive with respect to the action type of the interaction. In the case of two passive actions the resulting interaction will also be passive, i.e. rate 0. In the case of a passive and an active participant, the resulting interaction will also be active with the rate of the active action. Note that synchronisation between two immediate actions is not allowed.

This has implications for the compositionality since once an interaction involves one active participants only passive agents may be added later. When all possible interfaces to a model are restricted in this way a model is said to be *temporally closed*. Only temporally closed models are considered to be valid.

Another problem of this approach is that a renormalisation is necessary in order to get the correct rate when an active action is synchronised with two or more passive actions. This is currently carried out as a transformation of the labelled transition system which must be performed before the underlying Markov process can be extracted.

3 Evaluating the Definitions

In timed process algebras various properties have been identified and used to classify the different timed process algebras which have emerged [20]. These properties include:

Time determinism: For systems in which agents either witness an action or a time progression, it is assumed that if an agent witnesses a given time progression the outcome is unique.

Maximal progress: All interactions must occur before time progresses. One of the advantages of assuming maximal progress is that it makes it possible to construct agents in which particular actions are forced to happen.

Patience: An agent will idle until such time as it can synchronise. This assumption is not, in fact, adopted by many of the timed process algebras.

Persistence: If an agent is capable of performing a given action then the progression of time cannot remove the ability to perform that action. This is another unpopular assumption, although a form of it is adopted in [21].

It is natural to expect that similar properties or assumptions may be identified for stochastic process algebras. Such assumption are often not explicitly stated although they do exist. For example, during the development of PEPA two assumptions were made:

Bounded capacity: Each agent's description represents its capacity for carrying out work of a given type—this capacity is assumed to be bounded. In other words, the rate at which an agent carries out an action cannot be increased, so when it is involved in an interaction of the given type the rate of the interaction cannot be greater than the agent's individual rate for that action type.

Matched capacity: If two agents synchronise on an action which has the same representation and appears in a single instance in each of the interacting agents then the resulting interaction will have the same representation. In other words, if agents $(\alpha, r).P$ and $(\alpha, r).Q$ synchronise on actions of type α, then the resulting interaction will be (α, r).

The side condition imposed in B-MPA could also be regarded as an assumption:

Service interaction only: In each synchronisation only one agent is active with respect to the given action type, all other participants being passive. This corresponds to a service interaction.

As discussed above, this is only satisfied in B-MPA, although TIPP and PEPA allow such forms of interaction as a special case.

Clearly PEPA satisfies the assumptions of *bounded capacity* and *matched capacity* since they were used during the language development. It is interesting to note the extent to which these principles are satisfied, or not, by the other SPA languages.

TIPP satisfies neither assumption, since by definition an interaction will have a rate different from at least one of the participating actions, unless they both have rate 1. When both participating actions have rates greater than 1, both agents will appear to experience an enhanced capacity. When both participating actions have rates less than 1, both agents will appear to experience reduced capacity. When one participating action has a rate greater than 1 and one has a rate less than 1 they will experience reduced and enhanced capacity respectively.

The situation is not so clear in D-MPA, but if we assume that the number of instances of a action enabled in an agent represent the capacity to carry work of that type, then, as in TIPP, the assumption of bounded capacity is not satisfied. However, unlike TIPP, D-MPA does satisfy the assumption of *matched capacity*.

Since all interactions in B-MPA are restricted to involve at most one active participant, it follows that at most one participant represents a capacity to carry out work. Moreover, in effect this capacity is unaffected by the interactions that the agent may engage in. However, this is not ensured by the semantic rules of the language alone. When an active agent synchronises with an agent enabling two passive instances of the action type then, according to the semantic rules, the capacity to complete the work represented by this action is doubled. This is corrected by subsequent transformations of the labelled transition system which take place before the underlying Markov process is derived. Thus B-MPA satisfies the assumption of *bounded capacity* in its Markovian semantics but not in its structured operational semantics. Due to the restriction on the form of interactions in B-MPA, the principle of *matched capacity* is not relevant.

Let us consider again the situation when the principles of *bounded capacity* and *matched capacity* are not observed. Consider the following recursive definition of an agent:

$$\Big(rec\ Zeno : (\alpha, 2).Zeno \,\|_{\{\alpha\}}\, (\alpha, 2).Zeno\Big)$$

In TIPP this agent will perform an infinite sequence of α actions at progressively greater and greater rates as more copies of the $Zeno$ agent are introduced into the interaction. The term therefore represents an agent which has the capacity to complete an unbounded amount of work in a finite time. Conversely, if the rate of the α actions in the term was 1/2 instead of 2, the term would represent an agent which would perform α actions progressively slower at each unfolding of the recursion.

In D-MPA it is assumed that the rate at which the action α is completed remains constant at the rate μ_α. However, with each unfolding of the recursion more instances of the action become available to form the interaction, meaning that the capacity to carry out the action is increasing unboundedly.

In contrast, in PEPA the equivalent expression is the term:

$$Zeno \overset{\mathit{def}}{=} (\alpha, 2).Zeno \underset{\{\alpha\}}{\bowtie} (\alpha, 2).Zeno$$

This is an implicit recursion, but with each unfolding of the recursion the rate of the synchronised action remains the same, 2, due to the principle of matched capacity.

4 Interactions Between Systems

In this section we consider some interactions which take place between real systems, which may be classed as synchronisations using the definition introduced in Section 2. It is hoped that analysis of these interactions and the extent to which they can be captured by the combinators currently available in the SPA languages will increase our understanding of the features which must be included in languages to represent interactions between systems. Extracting common features of these interactions, and analysing distinctions, may also allow us to measure the usefulness of criteria, such as *matched capacity*, for comparing the expressiveness of systems.

4.1 Untimed Synchronisation

Perhaps the simplest interaction between agents which falls within our category of synchronisation is an untimed synchronisation. This is an instantaneous check-pointing action which ensures that both participants share some knowledge of their mutual situation. Each makes the other aware of its current state by participating in the synchronising action.

The most familiar notion of synchronisation is when agents agree on a common view of the current time. For example, during the course of a spy story, conspirators will often "synchronise watches".

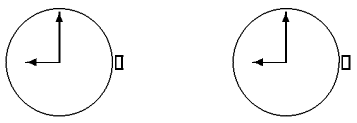
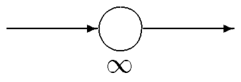


Figure 1: "Synchronise watches!"

Similarly in computer systems there is frequently a need for components to agree on the current time, or to be made aware of each other's current state, by experiencing a predetermined check-point action.

From an abstract perspective this can be viewed as the superposition of two instantaneous actions. One agent may be delayed, waiting for the other, but when both are ready they will instantly proceed to their next action. Thus the interaction will only become enabled when both are ready to participate.



Since B-MPA is the only SPA language to currently include immediate actions only B-MPA has the potential to represent this form of interaction. However, due to the assumption of *service interaction only*, this type of interaction can only be approximated by the synchronisation of one immediate action and one passive action.

4.2 Service

The notion of service is familiar to performance modellers from queueing networks where such interactions form the basis of all modelling. One participant provides a service or facility which is required by the other participant. Thus one agent is active with respect to the action, usually the server, whilst the other agent is passive, usually the client. Alternatively, the server may be regarded as a resource to be acquired, used and released, in which case it is natural to consider the client to be active whilst the server is passive. In either case there is an inherent asymmetry in the interaction since it is completely dominated by one of the participants.

If we consider a car and a petrol pump, the action of filling the car with petrol is an example of a service interaction. When the car is empty it cannot proceed further until its need for more petrol has been satisfied. However it cannot actively do anything further to achieve this except to make itself available for interaction with the petrol pump. When the pump is ready it will supply petrol at a rate it determines independently. When the tank is full the interaction is complete and both the car and the pump may proceed to their next action.

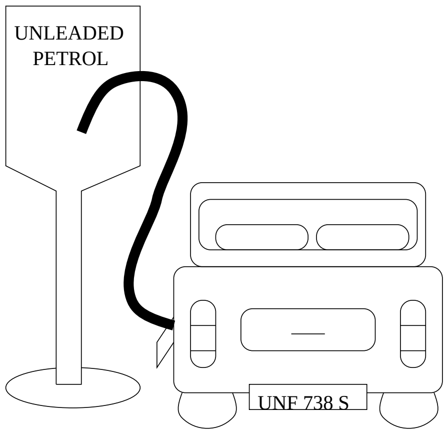
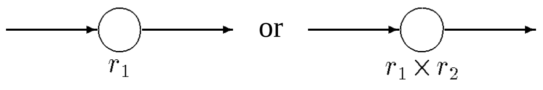


Figure 2: A service interaction between agents

This assumes that passive actions represent workload rather than working capacity. In TIPP this idea has been extended to incorporate scaling factors which represent an explicit service requirement relative to a standard service unit. In terms of the car and petrol pump example this is equivalent to the car assigning a rate to the filling action which is the ratio of the size of a standard tank to the size of its own tank. The petrol pump assigns a rate to the filling action which corresponds to the rate at which a standard tank is filled. When the car and the pump interact the rate of the filling action is adjusted according to the ratio presented by the car's representation of the action, as the rate at which the car is filled will be faster or slower according to whether the tank is smaller or larger than a standard tank. Note that the rate at which petrol is supplied is not changed: it is the rate at which the tank is filled which is dependent on the capacity of the tank.

The great success of queueing networks over the last 25 years testifies to the frequent occurrence of service interactions within computer and communication systems. For example, if a software system including a database is being modelled, a database `read` may be regarded as a service interaction between the application software and the database.

From an abstract perspective the service interaction can be regarded as the superposition of two actions: one active, representing working capacity, the other representing a workload or service requirement, *i.e. a passive action, or scaling factor. Neither agent can proceed until both are ready to participate. The interaction is then performed at the rate determined by the active participant, suitably modified to reflect the service requirement if scaling factors are being used.*



This scenario cannot be represented in D-MPA in which there are no passive actions or scaling factors. In PEPA and B-MPA straightforward service is modelled as an interaction between an active and a passive action. Indeed, all interactions in a B-MPA model must take this form if the model is to be temporally closed as required. Note that *immediate service* is also allowed (interaction of a passive and an immediate action) and may be used to represent interactions whose duration are negligible compared with other actions within the model. Scaling factors are only included in TIPP so that type of interaction, which we will term *flexible service* can only be captured in that language.

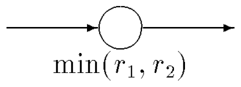
4.3 Patient Communication

The notion of patient communication is closely related to the principle of *bounded capacity*. In this case the interaction is assumed to represent a communication or shared task. The individual representations of each agent capture the rate at which that agent can complete the action necessary for the communication to take place. The interaction is completed by both agents working together *at the rate of the slower one*.

This scenario is illustrated by the case of two cyclists who wish to talk to each other as they cover a particular stretch of road. Each cyclist has his own rate of cycling determined by his bicycle, his fitness, etc. However in order to talk as they cycle they must cycle along side by side. This requires that the faster cyclist does not cycle as fast as he can but instead he adjusts his speed until it is the same as the other cyclist. This may involve a period of waiting for the other cyclist to catch up before the conversation can begin. Once their conversation is over they may again each go at their own pace.

Similarly, consider two computers communicating via a channel. Each component will have its own rate associated with the act of communication; for example, the rate at which it can prepare and submit packets to the channel, the bandwidth of the channel and the rate at which the receiver can collect and unpack packets. The rate at which the transfer can be completed will be bounded by the rate of the slowest component. Introducing a faster channel will not increase the rate at which the communication takes place if the rate at which packets can be received is the limiting factor. This form of interaction is clearly based on the assumption of *bounded capacity*: no component can be made to go faster by acting in parallel with another component. This is in contrast with the cooperation interaction which we will discuss in the next section.

From the abstract perspective this is again a superposition of actions. In contrast to service, we assume that both participants are active during the interaction. However, their individual actions are replaced by the interaction which has a rate which reflects the rate of the slower participant. The interaction is only enabled when both agents are ready to proceed and is not completed until they have both finished.



This form of interaction is the basis of the definition of "cooperation" used in PEPA. However, to overcome algebraic problems rather than the minimum of the rates of the individual actions involved in

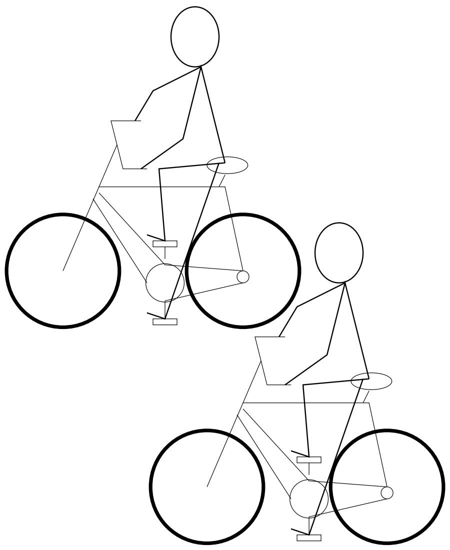


Figure 3: A patient communication interaction between agents

the interaction, the minimum of the apparent rates is assumed to be the apparent rate of the interaction with the actually rate adjusted according to the conditional probabilities of the individual actions involved. This form of interaction is not captured by any of the other SPA languages.

4.4 Cooperation

In some circumstances we can envisage interaction between agents which is less direct than the examples which have been considered so far. There are situations in which the presence of another agent carrying out the same action may influence how an agent completes an action although there is no direct interaction between them.

For example, consider a bicycle race. Each cyclist must rely on his own ability and as outlined in the previous example the speed at which he can proceed will be determined by his equipment, his fitness, etc. By cooperating, however, it is also possible for a group of cyclists to achieve speeds greater than they are individually capable of. By each taking a turn at the front of the pack, as well as turns in the slipstream of the other cyclists, the competitors cooperate to increase the total speed achieved by the group.

It is not apparent that there are any examples from computer or communication systems which illustrate this scenario. It assumes that the individual agents are aware of the other agents, although there is no direct interaction between them. Moreover they are able to vary their behaviour to respond to the situation. The increased speed is achieved because the cyclists are able to vary their rate of work, working harder when at the front of the pack, and less hard when at the back. Their total ability to work is perhaps unchanged but the rate at which they apply it is altered to respond to the situation.

In contrast with the other interactions which have been considered this does not seem to imply a superposition of actions. The individual actions remain. Moreover, there seems to be an implication that the action can be performed without interaction by any of the participants. However the interaction allows the rate of the action to be increased within all the participants.

As discussed in Section 2, all current SPA languages model synchronisations by superposition of actions.

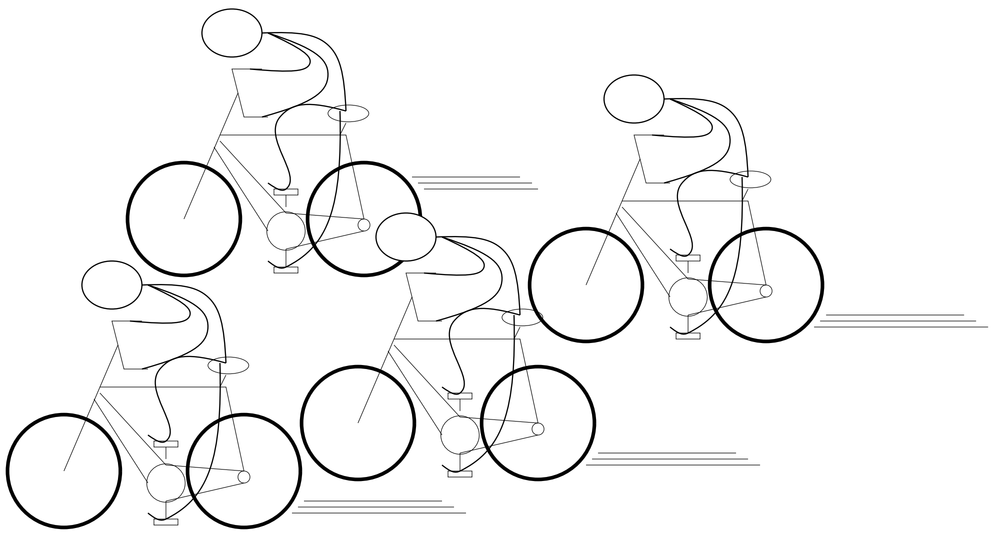


Figure 4: Cooperation between agents

The individual actions of agents are replaced by a new shared action with the rate $R = r_1 \otimes r_2$. Therefore this form of interaction cannot be captured by the current SPA languages.

In TIPP and D-MPA, the use of the product of rates for the operator $\otimes$ does reflect the idea that it is possible for agents to increase their capacity for work by working together. If both the initial rates were greater than 1 then the rate of the interaction will be greater than either of the rates of the individual actions which make it up. However, the value of the product is perhaps excessive. For example, if we consider two agents cooperating to complete an activity which individually they can undertake at rates 5 and 6 respectively, then the shared activity is conducted at rate 30. If this scenario is to be approximated by the superposition of actions it seems that a more appropriate operator $\otimes$ should be found.

In an SPA language agents which do not interact directly are modelled using the parallel combinator and an empty synchronisation or cooperation set. In this case, if the two agents are engaged in the same action at the same time the capacity of the combined component to carry out that action will be given as the sum of the capacities of the individual actions. However, this representation does not imply the mutual awareness of the two components which is implied within the cycling example. Nor is any increased capacity experienced by either of the participants.

4.5 Polite Communication

Previously we considered a patient form of communication between agents when it was assumed that each of the agents were active during the communication but possibly at a modified rate. In contrast we now consider communication between agents who behave subject to a polite protocol. The interaction or communication between the agents is viewed as a series of smaller interactions in which only one agent is active at a time. The emphasis is on exchange between the agents.

For example, consider two office workers. In the course of their daily routine they must make some telephone calls. Thus, when their behaviour is described each call is included, but only from the perspective of this agent. In other words, the duration of the call is represented to only reflect what this worker has to say during the call because he has no way of anticipating what the other caller will have to say. In order for the phone call to take place both parties must be ready to participate. Once the call commences the two speakers interleave their comments so that the time taken to complete the call is the sum of their individual durations for the call. Since they are polite there are no interruptions and they never both speak at the same time.

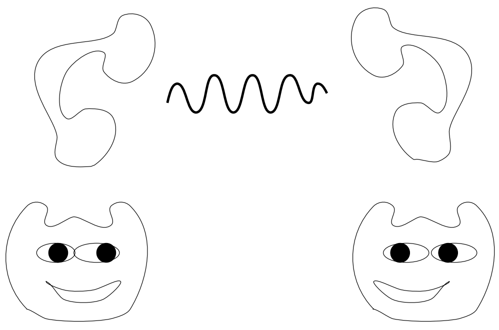
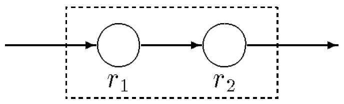


Figure 5: Communication between polite agents

An example of this scenario taken from the field of communication systems would be communication down a simplex channel. The channel can only support data flowing in one direction at a time. Thus the nodes at either end of the channel must alternate.

Here, again, the contributing actions are not superposed since we do not consider the two participants to be active at the same time. However, unlike the previous example we do assume that the interaction can only take place when both agents are ready to participate. Then the interleaving of parts of the individual actions is equivalent to the concatenation of the two actions. Thus, the interaction can be represented as a single action whose duration has a two phase Coxian distribution, one phase representing each of the contributing actions.



Since the current SPA languages are restricted to only consider actions with exponentially distributed durations, this type of interaction cannot be captured by any of the languages. In an early version of PEPA [22], this type of interaction was taken as the basis for the cooperation operator: shared actions were assigned the rate $R = (1/r_1 + 1/r_2)^{-1}$. Thus, although the duration was assumed to be exponentially distributed it was given the same mean as the associated Coxian distribution.

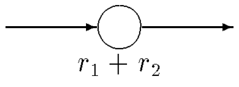
4.6 Impolite Communication

Not all communication between agents obeys the polite rules outlined in the previous section. It is not uncommon for interacting agents to both be active at the same time.

For example, consider again the office workers described in the previous section. As before we assume that there is a representation of the behaviour of each worker which includes all his tasks for the day, including any telephone calls which must be made. Also, calls cannot commence until both parties are ready to participate. However, once the call has started the callers do not behave in a polite way. Both immediately start talking and continue until they have finished everything they have to say or until the connection is severed. The first to finish speaking hangs up the call and terminates the interaction even though the other caller is still speaking.

This scenario is illustrated in a computer setting if we consider a node interacting with a slotted ring. Data cannot be transmitted by the node until it has access to a free slot. Transfer commences immediately. However, the amount of data which the node has to place on the ring, which might determine its own view of the interaction, is immaterial if it is greater than the slot available. Alternatively if the data is less than the slot the interaction will terminate in a shorter time than the ring might have anticipated.

From the abstract perspective this interaction represents the superposition of the individual actions of the two participants. Neither agent can commence until the other is also ready. However, once that occurs the interaction is loosely coupled as the two agents independently proceed with their own representation of the action. The first to finish its action terminates the interaction. This means that the duration of the interaction will be distributed as the minimum of the individual distributions. Since the individual distributions are exponentially distributed this means that the interaction will be exponentially distributed with the sum of the rates.



This scenario is not currently represented by any of the SPA languages. Although using the sum of the rates would be straightforward to calculate it does not satisfy the algebraic requirements outlined in Section 2. In particular addition is not distributive over addition, so it would not be possible to develop a congruence relation over such a combinator.

4.7 Timed Synchronisation

Finally we consider an interaction which aims to be a generalisation of the untimed synchronisation introduced at the beginning of this section. Here, agents work simultaneously towards a mutual end.

For example, consider two people eating a meal together in a restaurant (assume that they are polite!). Neither will start eating until both have been served their meal; they then both proceed to eat at their own rate until the food is finished. However, the meal is not considered to be over, and neither will leave the table, until both have finished eating.

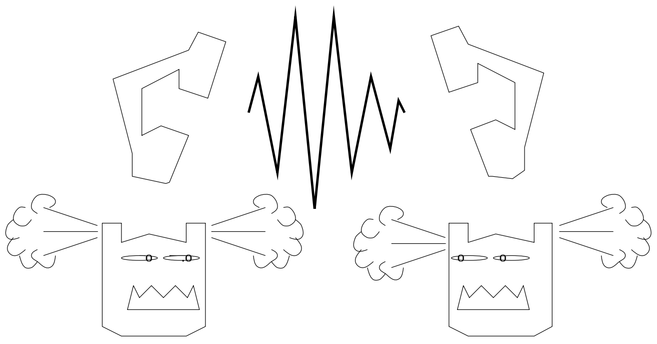


Figure 6: Communication between impolite agents

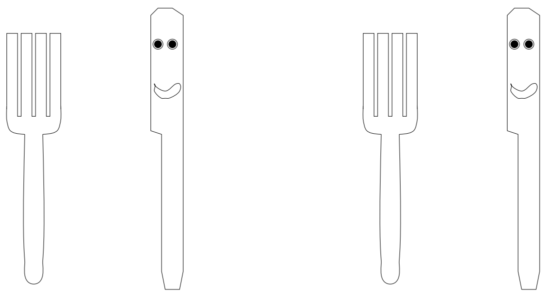
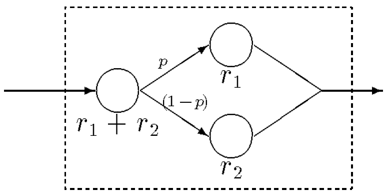


Figure 7: Timed synchronisation between agents

The computer example to illustrate this scenario is communication via a duplex connection. Now the channel can support simultaneous data flow in both directions so the nodes at either end of the channel can send data at the same time.

From the abstract perspective this interaction also represents a superposition of the actions of the two participants. Neither agent can commence until the other is also ready to participate. Once that occurs the interaction is loosely coupled as the two agents independently proceed with their own representation of the action. However the first to finish will wait for the other agent to also finish before the interaction is terminated. This means that the duration of the interaction will be distributed as the maximum of the individual distributions. Unfortunately the maximum of two exponential distribution is not another exponential distribution although it can be represented as a combination of such distributions, as shown below.



where $p = \dfrac{r_2}{r_1 + r_2}$

This scenario is also not currently represented by any of the SPA languages due to their restriction to exponential distributions.

5 Discussion and Conclusions

In Section 2 we saw that all the current SPA languages adopt the same form of interaction between agents. In all cases synchronisation is based on superposition of individual actions. Several different rules have been developed, however, for assigning a rate to the resulting interaction.

In Section 4 a variety of different possible interaction schemes were discussed based on the types of synchronisation, communication and interactions which occur between systems in the real world. This set of schemes is not exhaustive but it is sufficient to illustrate the many different ways in which interaction between systems can occur in a timed setting.

In view of this it has perhaps been ambitious of the stochastic process algebras to aim to define a single combinator to capture all synchronisations between agents. In fact, examination of the published case studies and examples, reveals that in most cases even the full expressibility of the defined combinator is not exploited. Instead, most of the models analysed in the literature are based on empty synchronisations when agents proceed completely independently, or alternatively on the restricted, *service*, form of interaction.

As remarked earlier service interaction is the basis of queueing networks which have been extensively used for performance modelling for more than 25 years with great success. The heavy reliance upon service interactions in SPA models has two possible explanations:

1. Service interaction is the basic form of interaction found in computer and communication systems and as such should be the primitive of any performance modelling paradigm.
2. Most of the published SPA models are based upon published performance studies or developed by people who have some experience of other performance modelling paradigms, in particular queueing networks. These modellers have been reluctant, or unable, to exploit the full SPA languages because they are unused to such expressibility

The real explanation is probably a mixture of both of the possibilities outlined above. Service interaction definitely occurs frequently in computer and communication systems. However, one of the motivations for considering new modelling paradigms has been the lack of expressibility of queueing networks. Care should be taken to avoid limiting the expressibility of SPA languages before the usefulness of other combinators has been fully explored.

Process algebras are sometimes regarded as a *cooperator* paradigm: the modelling style allows agents to be represented as cooperators and cooperands, as opposed to operators and operands. This style of modelling seems particularly appropriate to many modern distributed systems. Performance modellers using stochastic process algebras need time to make the mental switch necessary to fully exploit this new style of modelling.

Moreover, if service is to still play a fundamental rôle in SPA models, it would be appropriate to introduce an explicit combinator to represent it. It has already been demonstrated that it can be adequately represented using the parallel combinator. However, in the author's opinion there would be definite advantages in having a distinct combinator so that occurrences of service could be readily identified syntactically. Moreover, this would also allow the asymmetry of the interaction to be apparent at the syntactic level. A derived combinator could be defined, as in the case of the *subordination* combinator in CSP.

Similarly, the use of some actions to represent workloads, via the use of passive actions or scaling factor, should be given a clearer syntactic distinction. In some of the SPA languages, such as B-MPA, passive actions are assumed to always represent a workload requirement. However, in others, such as PEPA, it is intended that the passive actions will take a variety of rôles: a component may need to witness an action without actively taking part, it may have a rate which has not yet been specified, or it may be dominated by the other agent in any interaction (the service case). This suggests that merely having an action which is passive is not sufficient to identify workload.

Furthermore the rules for combining workloads will necessarily be different from the timed interactions between active components, or even the service interaction between a workload and an active component. However, currently there is no way to distinguish such cases. Such rules, for specifying the interactions between workloads, need to be defined.

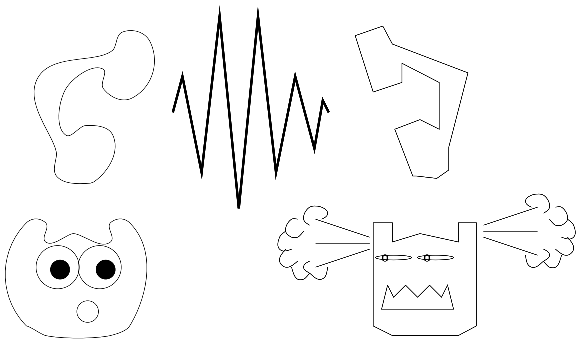


Figure 8: Asymmetric communication between polite and impolite agents

Finally, in addition to a new combinator specifically to represent service, it may be useful to provide other asymmetric combinators to capture unequal interactions between components. For example, consider the situation when an office worker who adheres to the polite protocol has to interact with an impolite office worker. The interaction will always be dominated by the impolite office worker although both parties are active and service is not involved.

Acknowledgements

This work was supported by a postdoctoral fellowship from the Engineering and Physical Sciences Research Council, B/93/RF/1729, and aided by a travel grant from The British Council and CNR, ROM/890/5A/62.

I would like to thank Gianfranco Balbo, Susanna Donatelli and Marina Ribaudo for the stimulating discussions from which this paper originated; Peter Buchholz for his help with some of the distributions; and Stephen Gilmore for several insightful comments and suggestions.

References

[1] H. Beilner, J. Maeter, and N. Weissenberg. Towards a Performance Modelling Environment: News on HIT. In R. Puigjaner, editor, *Proc. of 4th Int. Conf. on Modelling Techniques and Tools for Computer Peformance*. Plenum Press, 1988.

[2] K. Atif W.J. Stewart and B. Plateau. The Numerical Solution of Stochastic Automata Network. Technical Report 6, LMC-IMAG, November 1993.

[3] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[4] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[5] C-C. Jou and S.A. Smolka. Equivalences, Congruences and Complete Axiomatizations of Probabilistic Processes. In J.C.M. Baeten and J.W. Klop, editors, *CONCUR'90*, volume 458 of *LNCS*, pages 367–383. Springer-Verlag, August 1990.

[6] F. Moller and C. Tofts. A Temporal Calculus for Communicating Systems. In J.C.M. Baeten and J.W. Klop, editors, *CONCUR'90*, volume 458 of *LNCS*, pages 401–415. Springer-Verlag, August 1989.

[7] J. Davies and S. Schneider. A Brief History of Timed CSP. Technical report, Programming Research Group, Oxford University, Oxford OX1 3QD, September 1992.

[8] J. Baeten and J. Bergstra. Real Time Process Algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.

[9] U. Herzog. Formal description, time and performance analysis: A framework. Technical Report 15/90, IMMD VII, Friedrich-Alexander-Univeristät, Erlangen-Nürnberg, Germany, September 1990.

[10] N. Götz, U. Herzog, and M. Rettelbach. TIPP—a language for timed processes and performance evaluation. Technical Report 4/92, IMMD7, University of Erlangen-Nürnberg, Germany, November 1992.

[11] N. Götz, U. Herzog, and M. Rettelbach. Multiprocessor and Distributed System Design: The Integration of Functional Specification and Performance Analysis using Stochastic Process Algebras. In *Performance'93*, 1993.

[12] H. Hermanns and M. Rettelbach. Markovian Processes go Algebra. Technical Report 10/94, IMMD VII, University of Erlangen-Nürnberg, March 1994.

[13] S. Gilmore and J. Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In G. Haring and G. Kotsis, editors, *Proceedings of the Seventh International Conference on Modelling Technqiues and Tools for Computer Performance Evaluation*, volume 794 of *LNCS*, pages 353–368. Springer-Verlag, 1994.

[14] J. Hillston. *A Compositional Approach to Performance Modelling.* CST-107-94, Department of Computer Science, University of Edinburgh, April 1994.

[15] P. Buchholz. On a Markovian Process Algebra. Technical Report 500/194, Informatik IV, Universität Dortmund, April 1994.

[16] P. Buchholz. Compositional Analysis of a Markovian Process Algebra. In M. Rettelbach, editor, *Proc. of 2nd Process Algebra and Performance Modelling Workshop*, 1994.

[17] M. Bernardo, R. Gorrieri, and L. Donatello. MPA: A Stochastic Process Algebra. Technical Report UBLCS-94-10, Laboratory of Computer Science, University of Bologna, May 1994.

[18] M. Bernardo, R. Gorrieri, and L. Donatello. Describing Queueing Systems with MPA. Technical Report UBLCS-94-11, Laboratory of Computer Science, University of Bologna, May 1994.

[19] M. Bernardo, R. Gorrieri, and L. Donatello. Operational GSPN Semantics of MPA. Technical Report UBLCS-94-12, Laboratory of Computer Science, University of Bologna, May 1994.

[20] M. Hennessy. On timed process algebras: a tutorial. Technical Report 2/93, Department of Computer Science, University of Sussex, January 1993.

[21] Wang Yi. *A Calculus of Real Time Systems*. PhD thesis, Chalmers University, 1991.

[22] J. Hillston. PEPA - Performance Enhanced Process Algebra. Technical report, Dept. of Computer Science, University of Edinburgh, March 1993.