Programming Research Group

REPRESENTING NONDETERMINISM AND PROBABILISTIC BEHAVIOUR IN REACTIVE PROCESSES

Gavin Lowe

PRG-TR-11-93

Oxford University Computing Laboratory
11 Keble Road, Oxford OX1 3QD

Representing Nondeterminism and Probabilistic Behaviour in Reactive Processes*

Gavin Lowe

Abstract

In this paper we describe a way of modelling communicating processes that display both nondeterministic and probabilistic behaviour. We represent processes by *nondeterministic, probabilistic, action (NPA) graphs* with three sorts of nodes: action nodes, from which the process may evolve by performing visible actions; probabilistic nodes, from which the process may evolve probabilistically; and nondeterministic nodes, from which the process may evolve nondeterministically.

We define a number of types of "observation"—such as traces and refusals—that can be made of processes. For each type of observation, we describe how to abstract a corresponding denotational semantics from an NPA graph, so as to represent a process by a set of probability functions on observations—one function for each way of resolving the nondeterministic choices.

We illustrate our approach by using it to give a semantic model to a probabilistic version of CSP: we give an operational semantics for the language in terms of NPA graphs, and thus obtain a number of denotational semantic models, representing a process by a set of probability functions on observations.

*Further copies of this Technical Report may be obtained from the Librarian, Oxford University Computing Laboratory, Programming Research Group, 11 Keble Road, Oxford OX1 3QD, England (Telephone: +44-865-273837, Email: `library@comlab.ox.ac.uk`).

1 Introduction

In recent years, an important problem in the study of communicating systems has been the modelling of probabilistic behaviour. This is necessary if we are to argue formally about unreliable behaviour—for example that displayed by an unreliable communications medium—or probabilistic algorithms.

The usual approach has been to replace nondeterminism with probabilistic behaviour, by assigning probabilities to the different ways in which the process can act. However, I believe that it is important to be able to model *both* probabilities and nondeterminism, since these are different—although related—phenomena. Nondeterminism provides us with a powerful tool in the specification of systems as it opens up the possibility of underspecification, enabling us to leave certain aspects of a system's behaviour unspecified. For example, in the process[1]

$$a \longrightarrow b \longrightarrow P \sqcap b \longrightarrow a \longrightarrow P$$

(where the $\sqcap$ is a nondeterministic choice operator, and the $\longrightarrow$ is an action prefix operator) we are saying that we don't mind in what order the a and the b occur. However specifying that a process acts probabilistically is somewhat stronger; if we write down the process

$$a \longrightarrow b \longrightarrow P \;{}_{0.3}\!\oplus_{0.7}\; b \longrightarrow a \longrightarrow P$$

(where the ${}_{0.3}\!\oplus_{0.7}$ is a probabilistic choice operator that selects its first argument with probability 30%, and its second argument with probability 70%), then we are saying that over a large number of experiments we would expect to see the a performed first 30% of the time, and the b first 70% of the time. However, if we were to specify that we wanted the first process, and an implementor gives us the second, then we should not be upset, because the second process certainly meets the requirements specified by the first—in other words the second process *refines* the first.

Being able to represent nondeterminism in a probabilistic setting is also useful if we are to create a semantic model for a probabilistic language based upon CSP. The language of CSP contains many instances of nondeterminism; for example, the process

$$a \longrightarrow P \,\Box\, a \longrightarrow Q$$

(where the $\Box$ is an external choice operator that offers the environment a choice between the initial actions of its two arguments) will, after performing an a, nondeterministically act like either P or Q. In my thesis [Low93b] I got round this by assigning probabilities to the different sides of the external choice; for example, if the process

$$a \longrightarrow P \;{}_{0.4}\Box_{0.6}\; a \longrightarrow Q$$

performs an a, then with probability 40% it will be the one on the left, and with probability 60% it will be the one on the right. Similarly, the process

$$a \longrightarrow P \;{}_{0.4}\Box_{0.6}\; b \longrightarrow Q$$

will, in an environment that is willing to perform either an a or a b, perform the a with probability 40%, and the b with probability 60%. However the disadvantages of this approach

[1] In this paper we will be using a probabilistic language based upon CSP [Hoa85], which we will formally define in section 4. However, a similar analysis could be applied to any similar process algebra.

are that it forces us to over-specify processes, and it leads to a very complicated semantic model. It seems necessary to consider the "limit" of the probabilistic choice, ${}_{1}\Box_{0}$; this can be thought of as a prioritized choice, and thus our semantic model has to be able to deal with priorities. However, assigning probabilities to an external choice doesn't seem to gain us much: in most circumstances it is unnecessary to specify the way in which such choices are resolved: in other words, we allow the choice to be resolved nondeterministically.

In this paper we aim to develop semantic structures that allow us to argue about both nondeterminism and probabilistic behaviour. We then use these structures to produce a semantic model for a probabilistic language based upon CSP. The model will be used to formalize the notion of refinement described above.

Most previous semantic models for probabilistic behaviour have been operational in nature. We briefly review here what we believe to be some of the more important contributions.

Van Glabbeek et al. [vGSST90] discuss *reactive*, *generative* and *stratified* models of probabilistic processes.

- They define a *reactive* model to be one where the environment may only offer one event at a time. If the process can perform the offered event then it makes an internal state transition according to some probability distribution. For example, the process $\frac{1}{3}a.P + \frac{2}{3}a.Q + b.R$ will, after an a, act like P with probability $\frac{1}{3}$ and like Q with probability $\frac{2}{3}$.
- In a *generative* model, the environment can offer a choice between two or more events, and the process makes the choice according to some probability distribution. For example, if the process $\frac{1}{6}a.P + \frac{1}{3}a.Q + \frac{1}{2}b.R$ is offered an a or a b then it chooses the a with probability $\frac{1}{2}$ and the b with probability $\frac{1}{2}$; if the a is chosen, then the process acts like P with probability $\frac{1}{3}$, and like Q with probability $\frac{2}{3}$. If it is offered just an a, then the a will be performed (with probability 1), and it will then act like P with probability $\frac{1}{3}$, and like Q with probability $\frac{2}{3}$.
- A *stratified* model allows closer control over probabilistic choices. Consider a process that should choose an a with probability $\frac{1}{3}$, and otherwise choose between b and c with equal probabilities. The obvious definition of this is $P \mathrel{\widehat{=}} \frac{1}{3}a + \frac{1}{3}b + \frac{1}{3}c$. Consider however the case when the c is unavailable; with this definition, the a and the b are each chosen with probability $\frac{1}{2}$, rather than the desired $\frac{1}{3}$ and $\frac{2}{3}$. The process we require is $Q \mathrel{\widehat{=}} \frac{1}{3}a + \frac{2}{3}(\frac{1}{2}b + \frac{1}{2}c)$. If the c is unavailable then the a is chosen with probability $\frac{1}{3}$, and the b with probability $\frac{2}{3}$. The stratified model allows probabilistic choices between arbitrary processes.

Hans Hansson [Han91] has produced a discretely timed probabilistic process algebra based upon CCS, called TPCCS. Processes in his language alternate between probabilistic states (denoted by P, P', etc) and action states (denoted by N, N', etc). In action states, the process offers the environment a choice between a number of different actions; after performing an action, the process evolves into a probabilistic state: the environment is only allowed to offer one action at a time, so this is a reactive model in the terminology of [vGSST90]. In probabilistic states, processes evolve into action states according to some probability distribution. Thus, he differentiates between external and probabilistic choice.

In these works we can identify two different sorts of probabilistic branching which we term internal and external.

- With internal probabilistic branching, the process branches probabilistically without any reference to the environment, for example, as in the reactive and stratified models and Hansson's alternating model.
- With external probabilistic branching, the environment can offer several events, and the process can choose between them according to some probability distribution, as in the generative model.

External probabilistic choice can be used to model probabilistic internal branching by offering a choice between two internal τ events; for example $\frac{1}{3}\tau.P + \frac{2}{3}\tau.Q$ will act like P with probability *1/3* and like Q with probability *2/3*. However we do not believe that a probabilistic external choice is particularly useful in its own right, and can add considerably to the complexity of the semantic model: we therefore restrict ourselves to a probabilistic internal choice in this paper.

Previous attempts at producing denotational models for probabilistic languages have tried to represent a process as a function from "observations" to probabilities. One of the original aims of the research described in this paper was to produce a *compositional* semantics of this form for the language described above—i.e. where the semantics of a composite process is given in terms of the semantics of its subcomponents. However, as we will see, this is not possible for any of the obvious forms of "observation".

Karen Seidel [Sei92] has produced two probabilistic models of CSP. Her first model gives a semantics to her language in terms of probability measures on the space of infinite traces. She gives a semantics to a number of operators, but is unable to define external choice in this model because the probability of a trace occurring depends on the environment. In order to give a semantics to external choice she defines conditional probability measures. For all processes P, sets of traces A, and traces y, the expression $(\!| P |\!)(A, y)$ represents the probability that P performs a trace from A *given that* the environment is willing to perform the trace y (and nothing else). This allows her to give a semantics to external choice, but unfortunately, it is not possible to give a semantic definition for hiding in this way.

In my thesis [Low93b] I gave a semantic model that covered both internal and external probabilistic behaviour in a probabilistic version of Timed CSP. However, as described above, it was necessary to consider prioritized behaviour, and this resulted in a very complicated semantic model.

A common theme between these two denotational approaches is that we have both eliminated all nondeterminism except for that caused by a probabilistic choice operator: Seidel based her probabilistic languages upon underlying deterministic languages, whereas I refined several of the CSP operators so as to produce deterministic versions. The costs of these decisions are restricted expressiveness in Seidel's case, and added complexity in mine. I believe that by allowing nondeterministic behaviour, we will be able to get round these problems in this paper.

Our approach in this paper has many similarities with that of Hansson [Han91]. He represents processes as graphs with two sorts of nodes, action and probabilistic. We extend this to include a third sort of node, nondeterministic nodes, from which processes can evolve nondeterministically. These Nondeterministic-Probabilistic-Action (NPA) graphs can be used to represent processes, and have the useful feature of representing the three sorts of evolution separately. However, it is not obvious how to abstract a denotational model from them directly. We therefore introduce Probabilistic-Action (PA) graphs, and show how a single NPA graph can be represented by a set of PA graphs, one graph for each way of resolving the nondeterminism. We also show how a PA graph can be represented by a denotational object, namely a probability function on "observations"; thus a denotational semantics, representing a process as a set of probability functions, can be abstracted from an NPA graph. This will allow us to define the notions of equivalence and refinement between NPA graphs. By considering various sorts of "observations"—for example traces [Hoa85] and failures [BHR84]—we can define various different denotational models, and thus different equivalences.

We illustrate these ideas by applying them to a probabilistic language based upon CSP [Hoa85]. We describe how the language can be given an operational semantics in terms of NPA graphs, and thus how each process can be represented by a set of probability functions, one function for each way the nondeterministic choices can be resolved. Unfortunately, none of the equivalences induced turn out to be congruences, and thus we show that it is not possible to give a compositional semantics using these denotational models.

The paper is organized as follows. In section 2 we formally define PA graphs, and define a number of denotational semantic models, each representing a PA graph by a probability function on "observations" of some type. In section 3 we define NPA graphs, show how an NPA graph can be represented by a set of PA graphs, and hence by a set of probability functions. In section 4 we introduce our probabilistic language, and give it an operational semantics in terms of NPA graphs. We then use the ideas of the previous sections to define, in section 5, a number of denotational semantic models for the language, and thus define corresponding equivalence and refinement relations. In section 6 we sum up, make comparisons with other work, and give some pointers to future work.

2 Probability-action graphs

In this section, we give our formal definition of a Probability-Action graph. We define a PA graph to be a 5-tuple:

$$(State, \Sigma, \longrightarrow, \longmapsto, t_0)$$

where:

- $State$ is a set of states, ranged over by t, t', etc.;
- Σ is a set of *visible* events, ranged over by a, b, etc.;
- $\longrightarrow : State \times \Sigma \times State$ is an action transition relation;
- $\longmapsto : State \times [0, 1] \times State$ is a probability transition relation;
- $t_0 : State$ is an initial state;

satisfying a number of conditions, given below.

We will adopt the normal conventions, writing $t \xrightarrow{a} t'$ iff $(t, a, t') \in \longrightarrow$ and $t \xmapsto{p} t'$ iff $(t, p, t') \in \longmapsto$. Informally:

- $t \xrightarrow{a} t'$ means that from state t, the system can perform an a and move to state t';
- $t \xmapsto{p} t'$ means that from state t, the system can undergo a probabilistic transition, and with probability p move to state t';

We will also write $P \xrightarrow{a}$ for $\exists P' \bullet P \xrightarrow{a} P'$, and $P \longrightarrow$ for $\exists a \bullet P \xrightarrow{a}$, etc.

If $t \in \mathrm{dom} \longmapsto$ we say that t is a probabilistic state and write $\mathsf{Prob}\, t$. Otherwise we say that t is an action state and write $\mathsf{Action}\, t$; note that this latter possibility includes the case where t has no transitions: in this case we say that t is a deadlocked state. We will write $\mathsf{Inits}\, t$ for the set of events available from state t, i.e. $\{a \mid t \xrightarrow{a}\}$.

We insist that our graphs satisfy the following conditions:

1. $\mathrm{dom} \longrightarrow$ and $\mathrm{dom} \longmapsto$ are disjoint;
2. $t \xrightarrow{a} t' \wedge t \xrightarrow{a} t'' \;\Rightarrow\; t' = t''$;
3. $t \overset{p}{\longmapsto} t' \wedge t \overset{p'}{\longmapsto} t' \;\Rightarrow\; p = p'$;
4. for all t such that $\mathsf{Prob}\, t$, $\sum\{\!|p \mid t \overset{p}{\longmapsto} t' \wedge t' \in \mathit{State}|\!\} = \mathit{1}$;
5. There is no infinite sequence of states $t_0, t_1, \ldots$ such that for all i there is some p such that $t_i \overset{p}{\longmapsto} t_{i+1}$.

We write $\mathcal{PA}$ for the set of all PA graphs.

The conditions can be interpreted as follows:

1. Each state is either an action state or a probabilistic state.
2. No nondeterminism occurs at action states: state t cannot perform an a and evolve to two different states.
3. Given a pair of states, t and t', a process can make at most one probabilistic transition from t to t'; this is a notational convenience to avoid problems with processes such as $P\ {}_{0.5}\!\oplus_{0.5}\ P$ which appears to have two probability 0.5 transitions to P; we will get around this by ensuring that this process has a *single*, probability 1 transition to P. (An alternative to this method, which some workers have used, it to use a *multi*relation.)
4. The probabilities labelling the transitions from a probabilistic state add up to one.
5. The graph cannot diverge: it cannot make an infinite sequence of probabilistic transitions without performing a visible action. This is a simplifying assumption which we adopt because we do not want to have to worry about divergence in this paper; in a future paper we hope to include a consideration of divergence.

Our PA graphs differs from Hansson's alternating graphs as follows:

- Hansson insists that probabilistic and action states occur in strict alternation; we allow them to occur in any order.
- We insist that there is no nondeterminism at action nodes, whereas Hansson allows nondeterminism by, for example, allowing two transitions labelled with the same action.
- All our actions are visible events, whereas Hansson allows the silent event τ; we do not want τs because they cause nondeterminism.
- We have made the simplifying assumption of no divergence, whereas Hansson does not make this assumption, although he largely ignores the fact that τs are internal, and treats them the same as any other event.

2.1 Example

Throughout this paper we will define graphs via diagrams, as in figure 1.

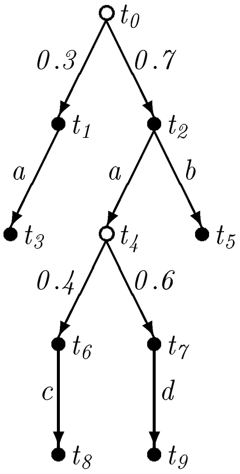

Figure 1: A PA graph

This represents the PA graph $(State, \Sigma, \longrightarrow, \longmapsto, t_0)$ with

$$\begin{aligned} State &= \{t_0, \ldots, t_9\} \\ \Sigma &= \{a, b, c, d\} \\ \longrightarrow &= \{(t_1, a, t_3),\ (t_2, a, t_4),\ (t_2, b, t_5),\ (t_6, c, t_8),\ (t_7, d, t_9)\} \\ \longmapsto &= \{(t_0, 0.3, t_1),\ (t_0, 0.7, t_2),\ (t_4, 0.4, t_6),\ (t_4, 0.6, t_7)\} \end{aligned}$$

The action states (t_1, t_2, t_3, t_5, t_6, t_7, t_8 and t_9) are represented by solid circles; the probabilistic states (t_0 and t_4) are represented by open circles.

2.2 Trace semantics for PA graphs

In order to give a denotational semantics and to define equivalences between processes, we will define a number of ways of abstracting a probability function from a PA graph. In this section we will describe how to obtain a probability function on traces from a PA graph. In later sections we will define other functions which abstract more information from PA graphs, and so make more distinctions between processes.

We define an observation in the traces model to be a sequence of events:

$$\mathcal{O}_{Tr} \mathrel{\widehat{=}} \Sigma^*$$

Then we can define a function

$$\mathcal{P}^{PA}_{Tr} : State \to \mathcal{O}_{Tr} \to [0, 1]$$

that, given a state in a PA graph returns a probability function on trace observations. Our interpretation of the function is as follows: given a sequence of events, suppose the environment offers these events one after another (but offers nothing else); what is the probability of the process accepting all of these events? Thus this is like the reactive model.

The function is defined as follows:

$$\mathcal{P}^{\mathcal{PA}}_{Tr}[\![t]\!]\langle\rangle = 1$$

$$\mathcal{P}^{\mathcal{PA}}_{Tr}[\![t]\!](a^\frown tr) = \begin{cases} 0 & \text{if } \mathsf{Action}\, t \wedge a \notin \mathsf{Inits}\, t \\ \mathcal{P}^{\mathcal{PA}}_{Tr}[\![t']\!]tr & \text{if } t \xrightarrow{a} t' \\ \sum\{\!|p \times \mathcal{P}^{\mathcal{PA}}_{Tr}[\![t']\!](a^\frown tr) \mid t \overset{p}{\longmapsto} t'|\!\} & \text{if } \mathsf{Prob}\, t \end{cases}$$

The empty trace can always occur with probability one. For non-empty traces of the form $a^\frown tr$, if we are at an action state from which an a is impossible, then this trace is obviously impossible, so is given probability 0; if an a is possible, then the probability is the probability of performing tr from the state reached after the a; if we are at a probabilistic state, then the probability is the weighted sum of the probabilities from all the successor states. The function is well defined because we have made the simplifying assumption that the graph does not diverge.

For example, consider state t_0 in the graph in figure 1; this has associated with it a probability function such that:

$$\langle\rangle \mapsto 1 \qquad \langle a\rangle \mapsto 1 \qquad \langle b\rangle \mapsto 0.7 \qquad \langle a, c\rangle \mapsto 0.28 \qquad \langle a, d\rangle \mapsto 0.42$$

and all other traces occur with probability 0.

2.3 Broom semantics for PA graphs

In this section we describe two other ways of abstracting information from a PA graph. Both are based upon the state reached after a particular trace: the first considers the acceptance set (the set of events that the process is willing to perform) in the state reached after the trace; the second is based upon refusal sets (sets of events none of which the process is able to perform).

Both of these consider the events available (or not available) only at the end of a trace: following Pnuelli [Pnu85] we term this *broom* semantics. In the next section we will consider denotations that also include information about the acceptances (refusals) at intermediate states: we term this *barb* semantics.

Acceptance sets

We define an observation in the broom, acceptance-set model to be a pair consisting of a sequence of events and a set of events:

$$\mathcal{O}_{BrA} \mathrel{\widehat{=}} \Sigma^* \times \mathbb{P}\,\Sigma$$

The interpretation of the pair (tr, X) is that the process can perform the trace tr, and end up in a state from which it is willing to perform any of the events from X (and no other events). More precisely, it will end up in a state from which the events of X could be performed, possibly after the performance of internal events.

We can define a function that, given a state in a PA graph, returns a probability function on broom, acceptance-set observations

$$\mathcal{P}^{\mathcal{PA}}_{BrA} : State \to \mathcal{O}_{BrA} \to [0, 1]$$

as follows:

$$\mathcal{P}^{\mathcal{PA}}_{BrA}[\![t]\!](\langle\rangle, X) = \begin{cases} 1 & \text{if } \mathsf{Action}\, t \wedge \mathsf{Inits}\, t = X \\ 0 & \text{if } \mathsf{Action}\, t \wedge \mathsf{Inits}\, t \neq X \\ \sum\{\!|\, p \times \mathcal{P}^{\mathcal{PA}}_{BrA}[\![t']\!](\langle\rangle, X) \mid t \overset{p}{\longmapsto} t' \,|\!\} & \text{if } \mathsf{Prob}\, t \end{cases}$$

$$\mathcal{P}^{\mathcal{PA}}_{BrA}[\![t]\!](a^\frown tr, X) = \begin{cases} \mathcal{P}^{\mathcal{PA}}_{BrA}[\![t']\!](tr, X) & \text{if } t \overset{a}{\longrightarrow} t' \\ 0 & \text{if } \mathsf{Action}\, t \wedge t \overset{a}{\not\longrightarrow} \\ \sum\{\!|\, p \times \mathcal{P}^{\mathcal{PA}}_{BrA}[\![t']\!](a^\frown tr, X) \mid t \overset{p}{\longmapsto} t' \,|\!\} & \text{if } \mathsf{Prob}\, t \end{cases}$$

If we are at an action node, then the probability of performing the empty trace and then having a particular acceptance set X is one precisely when X is the set of events possible from this state, and zero otherwise. If we are at a probabilistic state, then the probability is the weighted sum of the probabilities from the successor states. For non empty traces of the form $a^\frown tr$, if an a is possible from this state, then the probability is the probability for trace tr from the state reached after the a; if an a is impossible, then the probability is zero; if we are at a probabilistic state, then the probability is the weighted sum of the probabilities from the successor states.

Refusals

We define an observation in the broom, refusals model to be a pair consisting of a sequence of events and a set of events:

$$\mathcal{O}_{BrR} \mathrel{\widehat{=}} \Sigma^* \times \mathbb{P}\,\Sigma$$

The interpretation of the pair (tr, Y) is that the process can perform the trace tr, and end up in a state from which it is unwilling to perform any of the events from Y. We can define a function that, given a state in a PA graph, returns a probability function on broom, refusal observations

$$\mathcal{P}^{\mathcal{PA}}_{BrR} : \mathit{State} \rightarrow \mathcal{O}_{BrR} \rightarrow [0, 1]$$

as follows:

$$\mathcal{P}^{\mathcal{PA}}_{BrR}[\![t]\!](\langle\rangle, Y) = \begin{cases} 1 & \text{if } \mathsf{Action}\, t \wedge \mathsf{Inits}\, t \cap Y = \{\} \\ 0 & \text{if } \mathsf{Action}\, t \wedge \mathsf{Inits}\, t \cap Y \neq \{\} \\ \sum\{\!|\, p \times \mathcal{P}^{\mathcal{PA}}_{BrR}[\![t']\!](\langle\rangle, Y) \mid t \overset{p}{\longmapsto} t' \,|\!\} & \text{if } \mathsf{Prob}\, t \end{cases}$$

$$\mathcal{P}^{\mathcal{PA}}_{BrR}[\![t]\!](a^\frown tr, Y) = \begin{cases} \mathcal{P}^{\mathcal{PA}}_{BrR}[\![t']\!](tr, Y) & \text{if } t \overset{a}{\longrightarrow} t' \\ 0 & \text{if } \mathsf{Action}\, t \wedge t \overset{a}{\not\longrightarrow} \\ \sum\{\!|\, p \times \mathcal{P}^{\mathcal{PA}}_{BrR}[\![t']\!](a^\frown tr, Y) \mid t \overset{p}{\longmapsto} t' \,|\!\} & \text{if } \mathsf{Prob}\, t \end{cases}$$

If we are at an action node t, then the probability of performing the empty trace and then having a particular refusal set Y is one precisely when the graph cannot perform any of the events in Y, i.e. when $\mathsf{Inits}\, t \cap Y = \{\}$. Otherwise, the definition of $\mathcal{P}^{\mathcal{PA}}_{BrR}$ is the same as the definition of $\mathcal{P}^{\mathcal{PA}}_{BrA}$.

2.4 Barb semantics for PA graphs

In this section we investigate two denotational semantic models that give information about the events that are available throughout a trace.

Acceptance sets

We define the set $\mathcal{O}_{BaA}$ of barbed acceptance-set observations by

$$\mathcal{O}_{BaA} \mathrel{\widehat=} \{(tr, \Xi) : \Sigma^* \times (\mathbb{P}\,\Sigma)^* \mid \#\Xi = \#tr + 1 \wedge \forall\, i : \mathrm{dom}\, tr \bullet tr[i] \in \Xi[i]\}$$

(We write $s[i]$ for the ith element of sequence s, starting from 0, and $\mathrm{dom}\, s$ for the sequence $0 \mathrel{..} \#s - 1$.) A barbed acceptance-set observation is a trace tr of visible events performed and a sequence Ξ of acceptance sets. Informally, the pair

$$(\langle a_0, \ldots, a_{n-1}\rangle, \langle X_0, \ldots, X_n\rangle)$$

means that the process initially has acceptance set X_0, and performs a_0, then has acceptance set X_1, and performs a_1, and so on. The two conditions in the definition of $\mathcal{O}_{BaA}$ mean that there is one more acceptance set than visible event, and each event performed is a member of the corresponding acceptance set.

We can define a semantic function

$$\mathcal{P}^{\mathcal{PA}}_{BaA} : State \to \mathcal{O}_{BaA} \to [0, 1]$$

which returns a probability function on barbed acceptance-set observations:

$$\begin{aligned}
&\mathcal{P}^{\mathcal{PA}}_{BaA}[\![t]\!](\langle\rangle, \langle X\rangle) = \\
&\quad\begin{cases}
1 & \text{if } \mathsf{Action}\, t \wedge \mathsf{Inits}\, t = X \\
0 & \text{if } \mathsf{Action}\, t \wedge \mathsf{Inits}\, t \neq X \\
\sum\{\!|\, p \times \mathcal{P}^{\mathcal{PA}}_{BaA}[\![t']\!](\langle\rangle, \langle X\rangle) \mid t \overset{p}{\longmapsto} t' \,|\!\} & \text{if } \mathsf{Prob}\, t
\end{cases} \\
&\mathcal{P}^{\mathcal{PA}}_{BaA}[\![t]\!](a^\frown tr, X^\frown \Xi) = \\
&\quad\begin{cases}
\mathcal{P}^{\mathcal{PA}}_{BaA}[\![t']\!](tr, \Xi) & \text{if } t \overset{a}{\longrightarrow} t' \wedge \mathsf{Inits}\, t = X \\
0 & \text{if } \mathsf{Action}\, t \wedge (t \overset{a}{\not\longrightarrow} \vee \mathsf{Inits}\, t \neq X) \\
\sum\{\!|\, p \times \mathcal{P}^{\mathcal{PA}}_{BaA}[\![t']\!](a^\frown tr, X^\frown \Xi) \mid t \overset{p}{\longmapsto} t' \,|\!\} & \text{if } \mathsf{Prob}\, t
\end{cases}
\end{aligned}$$

The definition is very similar to the corresponding definition for broom semantics, except we record acceptance-set information throughout the observation.

Refusals

Similarly we can define the set of barbed refusal observations:

$$\mathcal{O}_{BaR} \mathrel{\widehat=} \{(tr, \Upsilon) : \Sigma^* \times (\mathbb{P}\,\Sigma)^* \mid \#\Upsilon = \#tr + 1 \wedge \forall\, i : \mathrm{dom}\, tr \bullet tr[i] \notin \Upsilon[i]\}$$

A barbed refusal observation is a trace tr of visible events performed and a sequence Υ of refusal sets. Informally, the pair $(\langle a_0, \ldots, a_{n-1}\rangle, \langle Y_0, \ldots, Y_n\rangle)$ means that the process can initially refuse Y_0, and performs a_0, then can refuse Y_1, and performs a_1, and so on. The two conditions in the definition of $\mathcal{O}_{BaR}$ mean that there is one more refusal set than visible event, and each event performed is not a member of the corresponding refusal set.

The following semantic function returns a probability function on barbed refusal observations.

$$\mathcal{P}^{\mathcal{PA}}_{BaR}[\![t]\!](\langle\rangle, \langle Y \rangle) =$$
$$\begin{cases} 1 & \text{if Action } t \wedge \text{Inits } t \cap Y = \{\} \\ 0 & \text{if Action } t \wedge \text{Inits } t \cap Y \neq \{\} \\ \sum\{\!\!\{ p \times \mathcal{P}^{\mathcal{PA}}_{BaR}[\![t']\!](\langle\rangle, \langle Y \rangle) \mid t \overset{p}{\longmapsto} t' \}\!\!\} & \text{if Prob } t \end{cases}$$
$$\mathcal{P}^{\mathcal{PA}}_{BaR}[\![t]\!](a^\frown tr, Y^\frown \Upsilon) =$$
$$\begin{cases} \mathcal{P}^{\mathcal{PA}}_{BaR}[\![t']\!](tr, \Upsilon) & \text{if } t \overset{a}{\longrightarrow} t' \wedge \text{Inits } t \cap Y = \{\} \\ 0 & \text{if Action } t \wedge (t \overset{a}{\not\longrightarrow} \vee \text{ Inits } t \cap Y \neq \{\}) \\ \sum\{\!\!\{ p \times \mathcal{P}^{\mathcal{PA}}_{BaR}[\![t']\!](a^\frown tr, Y^\frown \Upsilon) \mid t \overset{p}{\longmapsto} t' \}\!\!\} & \text{if Prob } t \end{cases}$$

2.5 Equivalences

For each of the five semantic functions we can define a corresponding equivalence between PA graphs by

$$t \equiv_i t' \;\Leftrightarrow\; \mathcal{P}^{\mathcal{PA}}_i[\![t]\!] = \mathcal{P}^{\mathcal{PA}}_i[\![t']\!] \qquad \text{for } i = Tr, BrA, BrR, BaA, BaR$$

It will turn out that the two broom equivalences identify precisely the same processes, as do the two barb equivalences. Also, the trace equivalence identifies more processes than the broom equivalences, which in turn identify more processes than the barb equivalences.

Note that

$$\mathcal{P}^{\mathcal{PA}}_{BrR}[\![t]\!](tr, Y) = \sum\{\!\!\{ \mathcal{P}^{\mathcal{PA}}_{BrA}[\![t]\!](tr, X) \mid Y \cap X = \{\} \}\!\!\}$$

(which we can prove by induction on the length of tr). Hence

$$t \equiv_{BrA} t' \;\Rightarrow\; t \equiv_{BrR} t'$$

Conversely, rearranging the above equation we have

$$\mathcal{P}^{\mathcal{PA}}_{BrA}[\![t]\!](tr, X) = \mathcal{P}^{\mathcal{PA}}_{BrR}[\![t]\!](tr, \Sigma - X) - \sum\{\!\!\{ \mathcal{P}^{\mathcal{PA}}_{BrA}[\![t]\!](tr, X') \mid X' \subset X \}\!\!\}$$

Hence, we can prove by induction on the size of X:

$$t \equiv_{BrR} t' \Rightarrow \forall\, tr, X \bullet \mathcal{P}^{\mathcal{PA}}_{BrA}[\![t]\!](tr, X) = \mathcal{P}^{\mathcal{PA}}_{BrA}[\![t']\!](tr, X)$$

So

$$t \equiv_{BrR} t' \;\Rightarrow\; t \equiv_{BrA} t'$$

Similarly

$$\mathcal{P}^{\mathcal{PA}}_{BaR}[\![t]\!](tr, \Upsilon) = \sum\{\!\!\{ \mathcal{P}^{\mathcal{PA}}_{BaA}[\![t]\!](tr, \Xi) \mid \forall\, i \bullet \Xi[i] \cap \Upsilon[i] = \{\} \}\!\!\}$$

(which we can prove by induction on the length of tr). Hence

$$t \equiv_{BaA} t' \;\Rightarrow\; t \equiv_{BaR} t'$$

Conversely, rearranging the above equation we have

$$\begin{aligned}&\mathcal{P}^{\mathcal{PA}}_{BaA}[\![t]\!](tr, \Xi) = \\ &\qquad \mathcal{P}^{\mathcal{PA}}_{BaR}[\![t]\!](tr, \langle \Sigma - \Xi[i] \mid i \in \operatorname{dom}\Xi\rangle) - \sum\left\{\!\!\left|\, \mathcal{P}^{\mathcal{PA}}_{BaA}[\![t]\!](tr, \Xi') \,\middle|\, \begin{array}{l}\forall\, i \in \operatorname{dom}\Xi \bullet \Xi'[i] \subseteq \Xi[i] \\ \wedge\ \Xi' \neq \Xi\end{array} \right|\!\!\right\}\end{aligned}$$

Hence, we can prove by induction on $\sum\{\!|\#(\Xi[i]) \mid i \in \operatorname{dom}\Xi|\!\}$:

$$t \equiv_{BaR} t' \Rightarrow \forall (tr, \Xi) : \mathcal{O}_{BaA} \bullet \mathcal{P}^{\mathcal{PA}}_{BaA}[\![t]\!](tr, \Xi) = \mathcal{P}^{\mathcal{PA}}_{BaA}[\![t']\!](tr, \Xi)$$

So

$$t \equiv_{BaR} t' \;\Rightarrow\; t \equiv_{BaA} t'$$

Also

$$\mathcal{P}^{\mathcal{PA}}_{Tr}[\![t]\!](tr) = \sum\{\!|\mathcal{P}^{\mathcal{PA}}_{BrA}[\![t]\!](tr, X) \mid X \in \mathbb{P}\,\Sigma|\!\}$$

So

$$t \equiv_{BrA} t' \Rightarrow t \equiv_{Tr} t'$$

And

$$\mathcal{P}^{\mathcal{PA}}_{BrA}[\![t]\!](tr, X) = \sum\{\!|\mathcal{P}^{\mathcal{PA}}_{BaA}[\![t]\!](tr, \Xi) \mid X = \mathsf{last}\,\Xi|\!\}$$

So

$$t \equiv_{BaA} t' \Rightarrow t \equiv_{BrA} t'$$

To summarize:

$$t \equiv_{BaA} t' \Leftrightarrow t \equiv_{BaR} t' \Rightarrow t \equiv_{BrA} t' \Leftrightarrow t \equiv_{BrR} t' \Rightarrow t \equiv_{Tr} t'$$

3 Nondeterministic-Probability-Action graphs

In this section we extend the concept of PA graphs in order to model nondeterminism. The basic idea is to allow a third type of node—nondeterministic nodes—from which the process can evolve nondeterministically.

We formally define an NPA graph to be a 6-tuple:

$$(State, \Sigma, \longrightarrow, \longmapsto, \multimap\!\!\!\!\rhd, s_0)$$

where:

- $State$ is a set of states, ranged over by s, s', etc.;
- Σ is a set of *visible* events, ranged over by a, b, etc.;
- $\longrightarrow : State \times \Sigma \times State$ is an action transition relation;
- $\longmapsto : State \times [0, 1] \times State$ is a probability transition relation;
- $\multimap\!\!\!\!\rhd : State \times State$ is a nondeterministic transition relation;
- $s_0 : State$ is an initial state;

satisfying a number of conditions, given below.

We will adopt the normal conventions, writing $s \xrightarrow{a} s'$ iff $(s, a, s') \in \longrightarrow$, $s \xmapsto{p} s'$ iff $(s, p, s') \in \longmapsto$, and $s \longrightarrow\!\!\triangleright\, s'$ iff $(s, s') \in \longrightarrow\!\!\triangleright$. Informally:

- $s \xrightarrow{a} s'$ means that from state s, the system can perform an a and move to state s';
- $s \xmapsto{p} s'$ means that from state s, the system can undergo a probabilistic transition, and with probability p move to state s';
- $s \longrightarrow\!\!\triangleright\, s'$ means that from state s, the system can undergo a nondeterministic transition and move to state s'.

We will also write $P \xrightarrow{a}$ for $\exists P' \bullet P \xrightarrow{a} P'$, and $P \longrightarrow$ for $\exists a \bullet P \xrightarrow{a}$, etc.

If $s \in \operatorname{dom} \longmapsto$ we say that s is a probabilistic state and write $\mathsf{Prob}\, s$. If $s \in \operatorname{dom} \longrightarrow\!\!\triangleright$ we say that s is a nondeterministic state and write $\mathsf{Nondet}\, s$. Otherwise we say that s is an action state and write $\mathsf{Action}\, s$; note that this last possibility includes the case where s has no transitions: in this case we say that s is a deadlocked state.

We insist that our graphs satisfy the following conditions:

1. $\operatorname{dom} \longrightarrow$, $\operatorname{dom} \longmapsto$ and $\operatorname{dom} \longrightarrow\!\!\triangleright$ are disjoint;
2. $s \xrightarrow{a} s' \wedge s \xrightarrow{a} s'' \;\Rightarrow\; s' = s''$;
3. $s \xmapsto{p} s' \wedge s \xmapsto{p'} s' \;\Rightarrow\; p = p'$;
4. for all s such that $\mathsf{Prob}\, s$, $\sum \{\!| p \mid s \xmapsto{p} s' \wedge s' \in \mathit{State} |\!\} = \mathit{1}$;
5. There is no infinite sequence of states $s_0, s_1, \ldots$ such that for all i, either there is some p such that $(s_i, p, s_{i+1}) \in \longmapsto$, or $(s_i, s_{i+1}) \in \longrightarrow\!\!\triangleright$.

We write $\mathcal{NPA}$ for the set of all NPA graphs satisfying these conditions.

These conditions can be interpreted as follows:

1. Each state is either an action state, a probabilistic state, or a nondeterministic state.
2. No nondeterminism occurs at action states; state s cannot perform an a and evolve to two different states.
3. Given a pair of states, s and s', a process can make at most one probabilistic transition from s to s'.
4. The probabilities labelling the transitions from a probabilistic state add up to one.
5. The graph cannot diverge: it cannot make an infinite sequence of state changes without performing a visible action.

3.1 Example

We will illustrate NPA graphs via diagrams, as in figure 2.

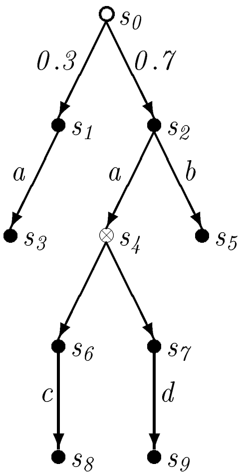

Figure 2: An NPA graph

This represents the NPA graph $(State, \Sigma, \longrightarrow, \longmapsto, \multimap, s_0)$ with

$$
\begin{aligned}
State &= \{s_0, \ldots, s_9\} \\
\Sigma &= \{a, b, c, d\} \\
\longrightarrow &= \{(s_1, a, s_3),\ (s_2, a, s_4),\ (s_2, b, s_5),\ (s_6, c, s_8),\ (s_7, d, s_9)\} \\
\longmapsto &= \{(s_0, 0.3, s_1),\ (s_0, 0.7, s_2)\} \\
\multimap &= \{(s_4, s_6),\ (s_4, s_7)\}
\end{aligned}
$$

As before, action states and probabilistic states are represented by solid and open circles, respectively. Nondeterministic states are represented by $\otimes$.

3.2 Factoring out nondeterminism

An NPA graph can be represented by a collection of PA graphs. We say that a state t in a PA graph is a direct refinement of a state s in an NPA graph, written $s \trianglelefteq t$, if the graph rooted at t can be obtained from the graph rooted at s by replacing each nondeterministic transition by a probabilistic transition with some probability.

Formally we define $\trianglelefteq$ to be the largest relation of type $\mathcal{NPA} \times \mathcal{PA}$ such that $s \trianglelefteq t$ iff

- $s \xrightarrow{a} s' \;\Rightarrow\; \exists\, t' : State \bullet t \xrightarrow{a} t' \wedge s' \trianglelefteq t'$;
- $t \xrightarrow{a} t' \;\Rightarrow\; \exists\, s' : State \bullet s \xrightarrow{a} s' \wedge s' \trianglelefteq t'$;
- $s \xmapsto{p} s' \;\Rightarrow\; \exists\, t' : State \bullet t \xmapsto{p} t' \wedge s' \trianglelefteq t'$;
- $s \multimap s' \;\Rightarrow\; \exists\, t' : State\, ; p : [0, 1] \bullet t \xmapsto{p} t' \wedge s' \trianglelefteq t'$;
- $t \xmapsto{p} t' \;\Rightarrow\; \exists\, s' : State \bullet (s \xmapsto{p} s' \vee s \multimap s') \wedge s' \trianglelefteq t'$.

Thus action nodes are refined by action nodes with the same initial actions, and probabilistic and nondeterministic nodes are both refined by probabilistic nodes.

For example, given the graphs in figures 1 and 2, we have that $s_0 \trianglelefteq t_0$; in fact we have $s_i \trianglelefteq t_i$ for $i = 0 \ldots 9$: the only interesting case to check is $s_4 \trianglelefteq t_4$, which follows from the fourth clause.

Note that we also have $s_0 \trianglelefteq t_0$ in figure 3 (where loops have been illustrated by giving several nodes the same label).

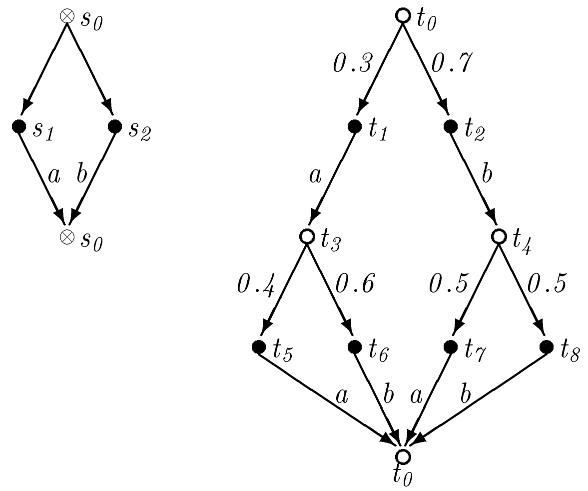

Figure 3: An example of direct refinement

In fact:

$$\begin{array}{ccc} s_0 \trianglelefteq t_0 & s_0 \trianglelefteq t_3 & s_0 \trianglelefteq t_4 \\ s_1 \trianglelefteq t_1 & s_1 \trianglelefteq t_5 & s_1 \trianglelefteq t_7 \\ s_2 \trianglelefteq t_2 & s_2 \trianglelefteq t_6 & s_2 \trianglelefteq t_8 \end{array}$$

In other words, a nondeterministic choice can be resolved with different probabilities each time round a loop.

We can now define the mapping $\mathsf{factor} : \mathcal{NPA} \to \mathbb{P}(\mathcal{PA})$, which maps an NPA graph to the set of its direct refinements by

$$\mathsf{factor}\, s \mathrel{\widehat{=}} \{t : \mathcal{PA} \mid s \trianglelefteq t\}$$

$\mathsf{factor}\, s$ contains all PA graphs obtained from s by replacing the nondeterministic nodes with probabilistic nodes.

We can use this to give a denotational semantics to NPA. Firstly we define five semantic spaces:

$$\mathcal{PM}_i \mathrel{\widehat{=}} \mathbb{P}(\mathcal{O}_i \to [0, 1]) \qquad \text{for } i = Tr, BrA, BrR, BaA, BaR$$

Each model represents an NPA graph by a set of probability functions on observations. We define semantic functions

$$\mathcal{P}_i^{\mathcal{NPA}} : \mathcal{NPA} \to \mathcal{PM}_i \qquad \text{for } i = Tr, BrA, BrR, BaA, BaR$$

by

$$\mathcal{P}_i^{\mathcal{NPA}}[\![s]\!] \mathrel{\widehat{=}} \{\mathcal{P}_i^{\mathcal{PA}}[\![t]\!] \mid t \in \mathsf{factor}\, s\} \qquad \text{for } i = Tr, BrA, BrR, BaA, BaR$$

The functions abstract from an NPA graph a set of probability functions, one function for each way of resolving the nondeterministic choices.

4 Probabilistic CSP

In this section we use NPA graphs to give a semantics to a probabilistic language based upon CSP.

4.1 Syntax

For simplicity we restrict ourselves to a fairly small syntax. We assume the existence of some set Σ of visible events. We define the language Probabilistic CSP (PCSP) by

$$P ::= a : A \longrightarrow P_a \mid P \sqcap P \mid P \,{}_p\!\oplus_q P \mid P \Box P \mid P \underset{A}{\parallel} P \mid f(P) \mid \mu X \bullet P$$

where A ranges over sets of events, f ranges over renaming functions of type $\Sigma \to \Sigma$, and p and q range over the interval $[0,1]$ with the property that $p + q = 1$.

The process $a : A \longrightarrow P_a$ is initially willing to perform any event a from A, and will then act like the corresponding process P_a. We will write $STOP$ for $a : \{\} \longrightarrow P_a$; this is the deadlocked process, which is unable to perform any events. We will write $a \longrightarrow P_a$ for $b : \{a\} \longrightarrow P_b$; this is the process that is initially willing to perform an a, and will then act like P_a.

$P \sqcap Q$ is the process that nondeterministically acts like either P or Q. $P \,{}_p\!\oplus_q Q$ will act like P with probability p, and like Q with probability q.

$P \Box Q$ offers the environment a choice between the initial actions of P and Q. If the environment chooses an event that only P can perform, then the process will continue to behave like P; if the environment chooses an event that only Q can perform, then the process will continue to behave like Q; if the environment chooses an event that both processes can perform then it is nondeterministic as to which process is chosen.

$P \underset{A}{\parallel} Q$ executes the processes P and Q in parallel, synchronizing on events from the set A. Events from outside A do not require synchronisation; if both sides can perform such an event then it is nondeterministic which process performs it.

The process $f(P)$ acts like P except all events are renamed by the function f. Nondeterminism can be introduced if two possible events are renamed to the same event.

The process $\mu X \bullet P$ acts like P but where all occurrences of the variable X represent recursive calls.

Note that we have not included a hiding operator. This is because there seems to be no way of giving a semantics to this operator within the context of NPA graphs. Consider the process $(a \longrightarrow b \longrightarrow STOP \Box c \longrightarrow STOP) \setminus a$. In the failures model, we have the law

$$(a \longrightarrow b \longrightarrow STOP \Box c \longrightarrow STOP) \setminus a = b \longrightarrow STOP \sqcap (b \longrightarrow STOP \Box c \longrightarrow STOP)$$

which has an obvious representation as an NPA graph. However this rule does not hold in more sophisticated models, such as the ready-set model of [OH86]. The point is that in the failures model, this instability can be modelled as nondeterminism, whereas this is not possible in other models. To get round this, it may be possible to model this instability in some other way, for example as in the refusal testing models of Phillips [Phi86] and Mukarram [Muk93]. However, the above syntax will be enough for our purposes.

4.2 Subsidiary functions

Before giving an operational semantics for our language in terms of NPA graphs, we need to define a few subsidiary functions. These are to ensure that our inference rules do not include negative premises.

Firstly, we define a function Action which returns true precisely when its argument should be represented by an action node. Then we define a function Inits which returns the set of events that the process is initially able to perform; our intention is that this should be empty if the process should be represented by a probabilistic or nondeterministic node.

The function Action is defined as follows:

$$\begin{aligned}
\mathsf{Action}(a : A \longrightarrow P_a) &\mathrel{\widehat{=}} \mathit{true} \\
\mathsf{Action}(P \sqcap Q) &\mathrel{\widehat{=}} \mathit{false} \\
\mathsf{Action}(P \;{}_p\!\oplus_q\; Q) &\mathrel{\widehat{=}} \mathit{false} \\
\mathsf{Action}(P \mathrel{\Box} Q) &\mathrel{\widehat{=}} \mathsf{Action}(P) \wedge \mathsf{Action}(Q) \\
\mathsf{Action}(P \mathrel{\underset{A}{\parallel}} Q) &\mathrel{\widehat{=}} \mathsf{Action}(P) \wedge \mathsf{Action}(Q) \\
\mathsf{Action}(f(P)) &\mathrel{\widehat{=}} \mathsf{Action}(P) \\
\mathsf{Action}(\mu\, X \bullet P) &\mathrel{\widehat{=}} \mathit{false}
\end{aligned}$$

Our intention with composite processes is that both components should finish making internal transitions before the composition can perform external actions: hence the form of the rules for external choice and parallel composition.

The function Inits is defined similarly:

$$\begin{aligned}
\mathsf{Inits}(a : A \longrightarrow P_a) &\mathrel{\widehat{=}} A \\
\mathsf{Inits}(P \sqcap Q) &\mathrel{\widehat{=}} \{\} \\
\mathsf{Inits}(P \;{}_p\!\oplus_q\; Q) &\mathrel{\widehat{=}} \{\} \\
\mathsf{Inits}(P \mathrel{\Box} Q) &\mathrel{\widehat{=}} \begin{cases} \mathsf{Inits}\, P \cup \mathsf{Inits}\, Q & \text{if } \mathsf{Action}\, P \wedge \mathsf{Action}\, Q \\ \{\} & \text{otherwise} \end{cases} \\
\mathsf{Inits}(P \mathrel{\underset{A}{\parallel}} Q) &\mathrel{\widehat{=}} \begin{cases} (\mathsf{Inits}\, P \cup \mathsf{Inits}\, Q) - A \cup (\mathsf{Inits}\, P \cap \mathsf{Inits}\, Q \cap A) & \\ & \text{if } \mathsf{Action}\, P \wedge \mathsf{Action}\, Q \\ \{\} & \text{otherwise} \end{cases} \\
\mathsf{Inits}(f(P)) &\mathrel{\widehat{=}} f(\mathsf{Inits}(P)) \\
\mathsf{Inits}(\mu\, X \bullet P) &\mathrel{\widehat{=}} \{\}
\end{aligned}$$

We have the following lemma, which can be proved by structural induction.

Lemma 4.1: $\mathsf{Inits}(P) \neq \{\} \Rightarrow \mathsf{Action}(P)$. ♡

$\mathsf{Inits}(P)$ is the set of events that P can initially perform; if this is non-empty then P should be represented by an action node. The converse is not true, because $\mathsf{Action}(STOP)$ but $\mathsf{Inits}(STOP) = \{\}$.

We now define functions Prob and Nondet which return true precisely when their arguments should be represented by probabilistic or nondeterministic nodes, respectively. Our intention is that probabilistic transitions should occur before nondeterministic transitions; for example, if P can evolve probabilistically and Q can evolve nondeterministically, then $P \mathbin{\Box} Q$ should undergo the probabilistic evolution first: this decision is fairly arbitrary in nature, but some such decision is needed in order to maintain the tripartition between action, probabilistic and nondeterministic nodes. Thus $P \mathbin{\Box} Q$ and $P \underset{A}{\parallel} Q$ should only evolve nondeterministically if neither P nor Q can evolve probabilistically.

$$
\begin{aligned}
\mathsf{Prob}(a : A \longrightarrow P_a) &\mathrel{\widehat{=}} \mathit{false} \\
\mathsf{Prob}(P \sqcap Q) &\mathrel{\widehat{=}} \mathit{false} \\
\mathsf{Prob}(P \,{}_p{\oplus}_q\, Q) &\mathrel{\widehat{=}} \mathit{true} \\
\mathsf{Prob}(P \mathbin{\Box} Q) &\mathrel{\widehat{=}} \mathsf{Prob}(P) \vee \mathsf{Prob}(Q) \\
\mathsf{Prob}(P \underset{A}{\parallel} Q) &\mathrel{\widehat{=}} \mathsf{Prob}(P) \vee \mathsf{Prob}(Q) \\
\mathsf{Prob}(f(P)) &\mathrel{\widehat{=}} \mathsf{Prob}(P) \\
\mathsf{Prob}(\mu\, X \bullet P) &\mathrel{\widehat{=}} \mathit{false}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{Nondet}(a : A \longrightarrow P_a) &\mathrel{\widehat{=}} \mathit{false} \\
\mathsf{Nondet}(P \sqcap Q) &\mathrel{\widehat{=}} \mathit{true} \\
\mathsf{Nondet}(P \,{}_p{\oplus}_q\, Q) &\mathrel{\widehat{=}} \mathit{false} \\
\mathsf{Nondet}(P \mathbin{\Box} Q) &\mathrel{\widehat{=}} (\mathsf{Nondet}(P) \vee \mathsf{Nondet}(Q)) \wedge \neg\, (\mathsf{Prob}(P) \vee \mathsf{Prob}(Q)) \\
\mathsf{Nondet}(P \underset{A}{\parallel} Q) &\mathrel{\widehat{=}} (\mathsf{Nondet}(P) \vee \mathsf{Nondet}(Q)) \wedge \neg\, (\mathsf{Prob}(P) \vee \mathsf{Prob}(Q)) \\
\mathsf{Nondet}(f(P)) &\mathrel{\widehat{=}} \mathsf{Nondet}(P) \\
\mathsf{Nondet}(\mu\, X \bullet P) &\mathrel{\widehat{=}} \mathit{true}
\end{aligned}
$$

The following lemma can be proved by structural induction:

Lemma 4.2: For each process P, precisely one of the following holds:

$$\mathsf{Action}(P) \qquad \mathsf{Prob}(P) \qquad \mathsf{Nondet}(P)$$

♡

Each process should either be represented by an action node, a probabilistic node, or a nondeterministic node.

4.3 Transition relations

We can now define operational rules for each of the constructs of the language.

Prefix choice

The process $a : A \longrightarrow P_a$ can perform any of the events from A, and then act like the corresponding process P_a.

$$a : A \longrightarrow P_a \;\xrightarrow{\;b\;}\; P_b \qquad (b \in A)$$

Nondeterministic choice

$P \sqcap Q$ can perform the obvious nondeterministic transitions:

$$P \sqcap Q \mathrel{-\!\!\rhd} P \qquad\qquad P \sqcap Q \mathrel{-\!\!\rhd} Q$$

Probabilistic choice

And $P \;{}_p\!\oplus_q\; Q$ can perform the obvious probabilistic transitions if P and Q are distinct:

$$\frac{}{P \;{}_p\!\oplus_q\; Q \xmapsto{p} P}\left[P \neq Q\right] \qquad\qquad \frac{}{P \;{}_p\!\oplus_q\; Q \xmapsto{q} Q}\left[P \neq Q\right]$$

where we are writing '$\neq$' for syntactic inequality. We also need the following rule:

$$P \;{}_p\!\oplus_q\; P \xmapsto{1} P$$

The above form of the rules is necessary to handle the process $P \;{}_{0.5}\!\oplus_{0.5}\; P$: the naïve form of these rules would give this process a single probability 0.5 transition to P.

External choice

$P \,\Box\, Q$ will perform probabilistic transitions if either of its components can:

$$\frac{\begin{array}{l} P \xmapsto{p} P' \\ \neg\ \mathsf{Prob}(Q) \end{array}}{P \,\Box\, Q \xmapsto{p} P' \,\Box\, Q} \qquad\qquad \frac{\begin{array}{l} Q \xmapsto{q} Q' \\ \neg\ \mathsf{Prob}(P) \end{array}}{P \,\Box\, Q \xmapsto{q} P \,\Box\, Q'}$$

$$\frac{\begin{array}{l} P \xmapsto{p} P' \\ Q \xmapsto{q} q' \end{array}}{P \,\Box\, Q \xmapsto{p \cdot q} P' \,\Box\, Q'}$$

(The precise form of these rules is necessary to ensure that from each probabilistic node the probabilities on transitions sum to one.)

If neither component can perform probabilistic transitions, but one can evolve nondeterministically, then that nondeterministic evolution will take place.

$$\frac{\begin{array}{l} P \mathrel{-\!\!\rhd} P' \\ \neg\ \mathsf{Prob}(Q) \end{array}}{P \,\Box\, Q \mathrel{-\!\!\rhd} P' \,\Box\, Q} \qquad\qquad \frac{\begin{array}{l} Q \mathrel{-\!\!\rhd} Q' \\ \neg\ \mathsf{Prob}(P) \end{array}}{P \,\Box\, Q \mathrel{-\!\!\rhd} P \,\Box\, Q'}$$

If one, and only one, of the subcomponents can perform a particular action, then that action is performed and the choice is resolved:

$$\frac{\begin{array}{l} P \xrightarrow{a} P' \\ \mathsf{Action}(Q) \\ a \notin \mathsf{Inits}(Q) \end{array}}{P \,\Box\, Q \xrightarrow{a} P'} \qquad\qquad \frac{\begin{array}{l} Q \xrightarrow{a} Q' \\ \mathsf{Action}(P) \\ a \notin \mathsf{Inits}(P) \end{array}}{P \,\Box\, Q \xrightarrow{a} Q'}$$

Note that this only happens when both components have finished internal probabilistic and nondeterministic transitions.

If both components can perform the same action a, then it is nondeterministic which one does the a.

$$\frac{\begin{array}{l} P \xrightarrow{a} P' \\ Q \xrightarrow{a} Q' \end{array}}{P \,\square\, Q \xrightarrow{a} P' \sqcap Q'}$$

Parallel composition

The rules for probabilistic and nondeterministic transitions for parallel compositions are similar to the corresponding rules for external choice:

$$\frac{\begin{array}{l} P \overset{p}{\longmapsto} P' \\ \neg\ \mathsf{Prob}(Q) \end{array}}{P \underset{A}{\parallel} Q \overset{p}{\longmapsto} P' \underset{A}{\parallel} Q} \qquad \frac{\begin{array}{l} Q \overset{q}{\longmapsto} Q' \\ \neg\ \mathsf{Prob}(P) \end{array}}{P \underset{A}{\parallel} Q \overset{q}{\longmapsto} P \underset{A}{\parallel} Q'}$$

$$\frac{\begin{array}{l} P \overset{p}{\longmapsto} P' \\ Q \overset{q}{\longmapsto} q' \end{array}}{P \underset{A}{\parallel} Q \overset{p \cdot q}{\longmapsto} P' \underset{A}{\parallel} Q'}$$

$$\frac{\begin{array}{l} P \mathrel{-\!\!\triangleright} P' \\ \neg\ \mathsf{Prob}(Q) \end{array}}{P \underset{A}{\parallel} Q \mathrel{-\!\!\triangleright} P' \underset{A}{\parallel} Q} \qquad \frac{\begin{array}{l} Q \mathrel{-\!\!\triangleright} Q' \\ \neg\ \mathsf{Prob}(P) \end{array}}{P \underset{A}{\parallel} Q \mathrel{-\!\!\triangleright} P \underset{A}{\parallel} Q'}$$

If one side of the parallel composition can perform an event from outside the synchronization set, then the parallel composition can:

$$\frac{\begin{array}{l} P \xrightarrow{b} P' \\ \mathsf{Action}(Q) \\ b \notin \mathsf{Inits}(Q) \end{array}}{P \underset{A}{\parallel} Q \xrightarrow{b} P' \underset{A}{\parallel} Q} \Big[\, b \notin A \,\Big] \qquad \frac{\begin{array}{l} Q \xrightarrow{b} Q' \\ \mathsf{Action}(P) \\ b \notin \mathsf{Inits}(P) \end{array}}{P \underset{A}{\parallel} Q \xrightarrow{b} P \underset{A}{\parallel} Q'} \Big[\, b \notin A \,\Big]$$

However, if both sides of the parallel composition can perform the same event $b \notin A$, then it is nondeterministic which side does the b.

$$\frac{\begin{array}{l} P \xrightarrow{b} P' \\ Q \xrightarrow{b} Q' \end{array}}{P \underset{A}{\parallel} Q \xrightarrow{b} \left(P' \underset{A}{\parallel} Q\right) \sqcap \left(P \underset{A}{\parallel} Q'\right)} \Big[\, b \notin A \,\Big]$$

Note that as with external choice, these transitions only take place when both sides are in action states.

The two components can synchronize on common events:

$$\frac{\begin{array}{l} P \xrightarrow{a} P' \\ Q \xrightarrow{a} Q' \end{array}}{P \underset{A}{\|} Q \xrightarrow{a} P' \underset{A}{\|} Q'} \Big[\, a \in A \,\Big]$$

Renaming

If P can perform nondeterministic or probabilistic transitions, then so can $f(P)$.

$$\frac{P \overset{p}{\longmapsto} P'}{f(P) \overset{p}{\longmapsto} f(P')} \qquad\qquad \frac{P \longrightarrow\!\!\triangleright\, P'}{f(P) \longrightarrow\!\!\triangleright\, f(P')}$$

If P can perform an event a, then $f(P)$ can perform $f(a)$. However, if P can perform several events b such that $f(a) = f(b)$ then it is nondeterministic which one will be done.

$$\frac{\begin{array}{l} \mathsf{Inits}\, P = B \\ \forall\, b \in B \bullet P \xrightarrow{b} P_b \end{array}}{f(P) \xrightarrow{f(a)} \bigsqcap_{\{b \mid f(a) = f(b)\}} P_b} \Big[\, a \in B \,\Big]$$

Recursion

Recursive processes can be unwound: we represent this unwinding by a nondeterministic transition:

$$\mu\, X \bullet P \longrightarrow\!\!\triangleright\, P[\mu\, X \bullet P / X]$$

4.4 A few lemmas

In this section we state a few lemmas about our operational semantics; each of these lemmas can be proved by structural induction. Firstly we show that the auxiliary functions Inits, Prob and Nondet meet their informal definitions.

$\mathsf{Inits}(P)$ is precisely those actions that P can perform initially.

Lemma 4.3: $a \in \mathsf{Inits}(P) \;\Leftrightarrow\; P \xrightarrow{a}$. ♡

$\mathsf{Prob}(P)$ is true of precisely those processes that can undergo probabilistic transitions.

Lemma 4.4: $\mathsf{Prob}(P) \;\Leftrightarrow\; P \longmapsto$. ♡

$\mathsf{Nondet}(P)$ is true of precisely those processes that can evolve nondeterministically.

Lemma 4.5: $\mathsf{Nondet}(P) \;\Leftrightarrow\; P \longrightarrow\!\!\triangleright$. ♡

We now show that the operational semantics we have defined meets the requirements of being an NPA graph.

The semantic definitions respect the tripartition of action, probabilistic and nondeterministic nodes.

Lemma 4.6: $\mathrm{dom} \longrightarrow$, $\mathrm{dom} \longmapsto$ and $\mathrm{dom} \longrightarrow\!\!\triangleright$ are disjoint. ♡

Nondeterminism does not occur at action nodes: processes cannot evolve via the same action into two different states.

Lemma 4.7: If $P \xrightarrow{a} P'$ and $P \xrightarrow{a} P''$ then $P' = P''$. ♡

The probabilities labelling transitions from a probabilistic node sum to one.

Lemma 4.8: If $\mathsf{Prob}(P)$ then $\sum\{\!|p \mid P \overset{p}{\longmapsto} P' \land P' \in State|\!\} = 1$. ♡

Recall the following condition on NPA graphs from section 3:

> There is no infinite sequence of states $s_0, s_1, \ldots$ such that for all i, either there is some p such that $s_i \overset{p}{\longmapsto} s_{i+1}$, or $s_i \rightarrow\!\!\triangleright\, s_{i+1}$.

This will only be true of a subset of our syntax: for example, consider the process $\mu X \bullet X$; our rules give this a nondeterministic transition to itself:

$$\mu X \bullet X \;\rightarrow\!\!\triangleright\; \mu X \bullet X$$

Thus this graph does not meet the above rule. To get around this, we restrict our attention to the non-divergent subset of the syntax. In a future paper we hope to extend our ideas in order to include a consideration of divergence.

5 Denotational semantics for PCSP

In this section we develop denotational semantic models for PCSP. We extend each of the ways of abstracting denotational information from a PA graph—described in sections 2.2, 2.3 and 2.4—to Probabilistic CSP.

Firstly we define a semantic function

$$\mathcal{G}_{\mathcal{NPA}} : PCSP \to \mathcal{NPA}$$

such that $\mathcal{G}_{\mathcal{NPA}}[\![P]\!]$ gives the NPA graph for P defined by the rules in section 4.

To extend the semantic functions to Probabilistic CSP we define functions

$$\mathcal{P}_i : PCSP \to \mathcal{PM}_i \qquad \text{for } i = Tr, BrA, BrR, BaA, BaR$$

by

$$\mathcal{P}_i[\![P]\!] \mathrel{\widehat{=}} \mathcal{P}_i^{\mathcal{NPA}} \circ \mathcal{G}_{\mathcal{NPA}} \qquad \text{for } i = Tr, BrA, BrR, BaA, BaR$$

$\mathcal{G}_{\mathcal{NPA}}$ builds the graph for the process; as described in section 3.2, $\mathcal{P}_i^{\mathcal{NPA}}$ abstracts the probability functions.

We can use these semantic functions to define notions of equivalence and refinement between processes. We define two processes to be *equivalent* (in a particular model) if they have the same probability functions:

$$P \equiv_i Q \Leftrightarrow \mathcal{P}_i[\![P]\!] = \mathcal{P}_i[\![Q]\!] \qquad \text{for } i = Tr, BrA, BrR, BaA, BaR$$

We say that Q is a *refinement* of P if every probability function for Q is a probability function for P:

$$P \sqsubseteq_i Q \Leftrightarrow \mathcal{P}_i[\![P]\!] \supseteq \mathcal{P}_i[\![Q]\!] \qquad \text{for } i = Tr, BrA, BrR, BaA, BaR$$

This corresponds to Q being more deterministic than P.

Note that the equivalences $\equiv_{BrA}$ and $\equiv_{BrR}$ identify the same processes, as do the equivalences $\equiv_{BaA}$ and $\equiv_{BaR}$, because the same is true of the corresponding equivalences over PA graphs.

5.1 Examples of refinement

As an example of the refinement relation, consider the process $P \;_p\!\oplus_q\; Q$. The NPA graph for this process is as in figure 4, where G_P and G_Q are the NPA graphs for P and Q, respectively.

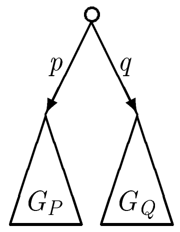

Figure 4: The NPA graph for $P \;_p\!\oplus_q\; Q$

Hence, each PA graph for $P \;_p\!\oplus_q\; Q$ is as in figure 5, where G'_P is a PA graph for P (so $G_P \trianglelefteq G'_P$) and G'_Q is a PA graph for Q (so $G_Q \trianglelefteq G'_Q$).

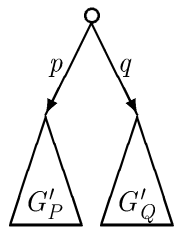

Figure 5: A PA graph for $P \;_p\!\oplus_q\; Q$

Let f_P be the probability function corresponding to G'_P, and let f_Q be the probability function corresponding to G'_Q (in one of the models). Then the probability of observation obs being accepted by this graph is $p.f_P(obs) + q.f_Q(obs)$. Hence the semantic representation of $P \;_p\!\oplus_q\; Q$ satisfies the following equation:

$$\mathcal{P}_i[\![P \;_p\!\oplus_q\; Q]\!] = \{\{obs \mapsto p.f_P(obs) + q.f_Q(obs)\} \mid f_P \in \mathcal{P}_i[\![P]\!] \wedge f_Q \in \mathcal{P}_i[\![Q]\!]\}$$

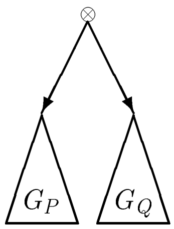

Figure 6: The NPA graph for $P \sqcap Q$

Now consider the process $P \sqcap Q$. This has an NPA graph as in figure 6, where G_P and G_Q are the NPA graphs for P and Q, respectively.

Hence all its PA graphs are as in figure 5, where $p, q \in [0, 1]$ with $p + q = 1$, and $G_P \trianglelefteq G'_P$, $G_Q \trianglelefteq G'_Q$. So the semantic representation of $P \sqcap Q$ satisfies the following equation:

$$\mathcal{P}_i[\![P \sqcap Q]\!] = \{\{obs \mapsto p.f_P(obs) + (1 - p).f_Q(obs)\} \mid f_P \in \mathcal{P}_i[\![P]\!] \wedge f_Q \in \mathcal{P}_i[\![Q]\!] \wedge p \in [0, 1]\}$$

Thus we see immediately that

$$P \sqcap Q \sqsubseteq_i P \;{}_p\!\oplus_q\; Q$$

So nondeterministic choices are refined by probabilistic choices, as we wanted.

Also, by considering the case $p = 1$ in the semantic equation for nondeterministic choice, we see that

$$P \sqcap Q \sqsubseteq_i P$$

and similarly

$$P \sqcap Q \sqsubseteq_i Q$$

as in the traditional CSP models.

5.2 Compositionality

Unfortunately, none of the equivalences is a congruence. Let

$$\begin{aligned} P \;\widehat{=}\;& (a \longrightarrow c \longrightarrow STOP \,\Box\, b \longrightarrow d \longrightarrow STOP) \\ & {}_{1/2}\!\oplus_{1/2} \\ & (a \longrightarrow e \longrightarrow STOP \,\Box\, b \longrightarrow f \longrightarrow STOP) \\ Q \;\widehat{=}\;& (a \longrightarrow c \longrightarrow STOP \,\Box\, b \longrightarrow f \longrightarrow STOP) \\ & {}_{1/2}\!\oplus_{1/2} \\ & (a \longrightarrow e \longrightarrow STOP \,\Box\, b \longrightarrow d \longrightarrow STOP) \end{aligned}$$

The NPA graphs for these processes are given in figure 7.

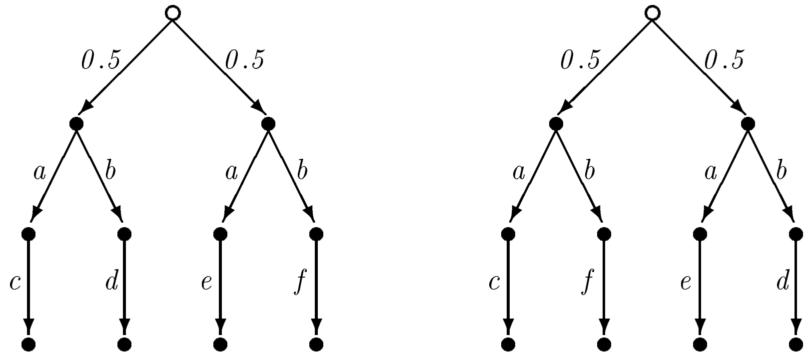

Figure 7: The processes P and Q

Then it is easily verified that $P \equiv Q$ for each of the equivalences. However, consider the processes $f(P)$ and $f(Q)$ where f maps b to a, and otherwise is the identity function. The NPA graphs for these processes are shown in figure 8.

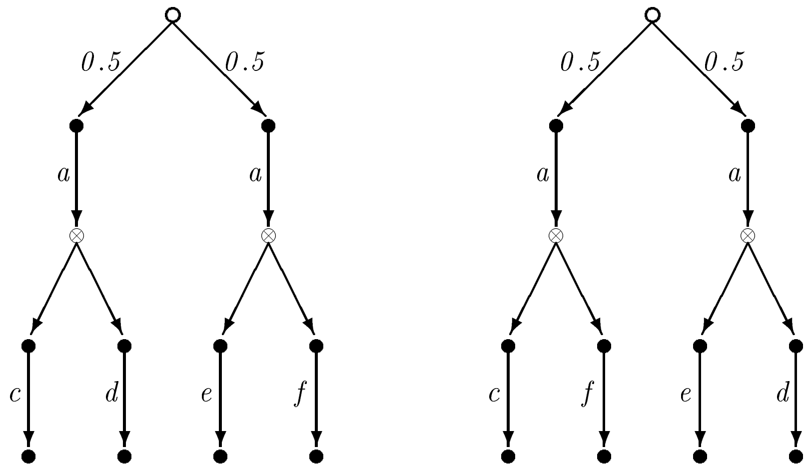

Figure 8: The processes $f(P)$ and $f(Q)$

Thus $f(P) \not\equiv_{Tr} f(Q)$, since one of the probability functions for $f(P)$ has

$$\langle a, c\rangle \mapsto 1/2 \qquad \langle a, f\rangle \mapsto 1/2 \qquad \langle a, d\rangle \mapsto 0 \qquad \langle a, e\rangle \mapsto 0$$

whereas none of the probability functions for $f(Q)$ is like this. Similarly, these processes are not equivalent under any of the other equivalences. Thus none of the equivalences is a congruence.

Normally, CSP-style languages are given a compositional denotational semantics, where the semantics of a composite process is given in terms of the semantics of its subcomponents. However, the fact that none of the equivalences is a congruence means that this is not possible in these models.

6 Conclusions

In this paper we have developed a way of representing communicating processes that display both probabilistic and nondeterministic behaviour. We have represented processes as graphs, called Nondeterministic-Probabilistic-Action (NPA) graphs, with three sorts of nodes: nondeterministic nodes from which processes evolve nondeterministically; probabilistic nodes, from which processes evolve according to some probability distribution; and action nodes, from which processes can evolve by performing visible actions. We have shown how a single NPA graph can be represented by a set of PA graphs with only probabilistic and action nodes. We have defined five notions of "observation", and have shown how each PA graph can have identified with it a probability function on such observations. Thus we have identified with each NPA graph a set of probability functions, one function for each way of resolving the nondeterminism.

We have illustrated NPA graphs by using them to give an operational semantics to a probabilistic version of CSP. From this we have abstracted five denotational semantic models, representing a process by a set of probability functions, and thus defined notions of probabilistic equivalence and refinement. We have shown that, unlike in the unprobabilistic case, the equivalences induced by considering refusals is the same as from acceptance sets. Unfortunately, none of the equivalences is a congruence, and thus we are unable to produce a compositional semantics of this form.

6.1 Comparisons

Jou and Smolka [JS90] have considered various equivalences for probabilistic processes using the generative model, i.e. with external probabilistic choices. They extend the notions of trace [Hoa85], failure [BHR84], ready [OH86], and bisimulation [Mil89] equivalence to the probabilistic case. They show that—as in our case—there is no difference between failures and ready equivalence, and that trace and failure equivalences are not congruences. These are interesting results because—despite using a very different setting to us—they obtain essentially the same results.

I do not believe that bisimulation equivalence is the equivalence that we want, because it makes more distinctions than we would really like: for example, it distinguishes between the two processes in figure 9, which we would hope to identify because we want to abstract away from the points at which choices are made.

Jou and Smolka's work is based upon broom tests. Christoff [Chr90] extends this to barb tests and defines a number of equivalences based on these tests. However I don't think it is possible to adapt these equivalences to NPA graphs, because they depend heavily upon the generative nature of his processes, using probabilities on external choices.

6.2 Future work

The results we have obtained in this paper are—I think—very disappointing. Our aim has been to produce a compositional semantics for a probabilistic language based upon CSP, but we have shown that this is not possible, at least in any of the obvious settings. It may be possible to find some notion of equivalence which is a congruence—for example, the largest congruence contained in trace equivalence—but this equivalence will have to be very fine, distinguishing, for example, between the two processes in figure 7.

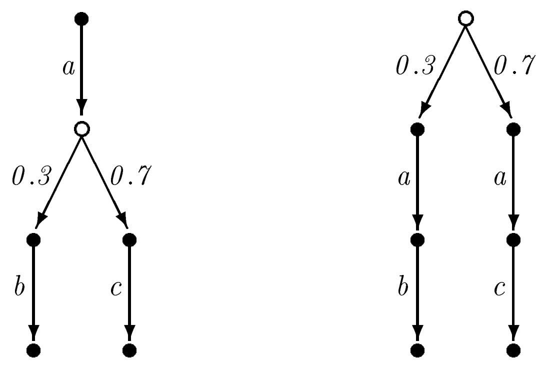

Figure 9: Two processes that are not bisimilar

However, I believe that the methods developed will prove useful in the verification of probabilistic systems. I am currently developing a tool—Pravda [Low93a]—which tests for refinement between processes, as formally defined in this paper. I believe that this gives an intuitively clear way of specifying probabilistic processes. For example, consider the process

$$P \mathrel{\widehat{=}} a \longrightarrow STOP \;{}_{0.9}\!\oplus_{0.1}\; tick \longrightarrow P$$

Here the a is supposed to represent some desirable outcome, and the *tick* is supposed to represent some time passing: each second, P will perform an a with probability 90%. Suppose we want to test whether P will perform an a within 2 seconds with probability at least 99%; then we have to test whether P refines the most nondeterministic process that meets this informal specification, namely

$$Q \mathrel{\widehat{=}} (a \longrightarrow STOP \sqcap tick \longrightarrow a \longrightarrow STOP) \;{}_{0.99}\!\oplus_{0.01}\; CHAOS$$

where $CHAOS$ is the most nondeterministic process:

$$CHAOS \mathrel{\widehat{=}} a \longrightarrow CHAOS \sqcap tick \longrightarrow CHAOS \sqcap STOP$$

6.3 Acknowledgements

I has benefited from discussions with Ivan Christoff and Bill Roscoe about probabilistic processes. Brian Scott and Steve Schneider both made useful comments about the content and presentation of this paper.

This paper was written while I was employed on the Esprit BRA project REACT.

References

[BHR84] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of CSP. *Journal of the ACM*, 31(3):560–599, 1984.

[Chr90] Ivan Christoff. Testing equivalences for probabilistic processes. Technical Report DoCS 90/22, Uppsala University, Sweden, 1990.

[Han91] Hans A. Hansson. *Time and Probability in Formal Design of Distributed Systems.* PhD thesis, Swedish Institute of Computer Science, 1991. Published as SICS Dissertation Series, number 05.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[JS90] Chi-Chang Jou and Scott A. Smolka. Equivalences, congruences and complete axiomatizations for probabilistic processes. In *Concur '90, LNCS 458*, 1990.

[Low93a] Gavin Lowe. Pravda: a tool for verifying probabilistic processes. In Jane Hillston and Faron Moller, editors, *Proceedings of the Workshop on Process Algebra and Performance Modelling*, 1993. University of Edinburgh report CSR-26-93.

[Low93b] Gavin Lowe. *Probabilities and Priorities in Timed CSP.* D. Phil thesis, Oxford, 1993.

[Mil89] Robin Milner. *Communication and Concurrency.* Prentice Hall International, 1989.

[Muk93] Abida Mukarram. *A Refusal Testing Model for CSP.* D. Phil thesis, Oxford, 1993.

[OH86] E. R. Olderog and C. A. R. Hoare. Specification-oriented semantics for communicating processes. *Acta Informatica*, 23(1):9–66, 1986.

[Phi86] Iain Phillips. Refusal testing. In *Proceedings of 13th International Colloquium on Automata, Languages and Programming, LNCS 226*, pages 304–313. Springer Verlag, 1986.

[Pnu85] Amir Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *Proceedings of 12th International Colloquium on Automata, Languages and Programming, LNCS 194*, pages 15–32. Springer Verlag, 1985.

[Sei92] Karen Seidel. *Probabilistic Communicating Processes.* D. Phil thesis, Oxford, 1992.

[vGSST90] R. J. van Glabbeek, S. A. Smolka, B. Steffen, and C. Tofts. Reactive, generative and stratified models of probabilistic processes. In *IEEE Symposium on Logic in Computer Science*, 1990.