LFCS

Laboratory for Foundations of Computer Science
Department of Computer Science - University of Edinburgh

Relative Frequency in a Synchronous Calculus

by

Chris Tofts

Relative Frequency in a Synchronous Calculus

LFCS Report Series

ECS-LFCS-90-108

LFCS
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

February 1990

Copyright © 1990, LFCS

Relative Frequency in a Synchronous Calculus.

Chris Tofts

February 28, 1990

Abstract

We present a weighted synchronous calculus that can be interpreted as reasoning over probabilistic processes [LS89,SST89,GSST90]. The abstraction from absolute weight to relative frequency is obtained semantically. We also add a notion of dominance which can be interpreted as priority. This notion is shown to be dual to that of "zero probability" [SST89,GSST90] and can be used to construct arbitrary priority structures. Finally, an equational system for reasoning about the weighted processes is presented.

1 Introduction.

The properties of both sequential [Wel83,Rab76] and concurrent [CJS90,LS89,GSST90,SST89] probabilistic computation are interesting for several reasons. The notion of probability was developed to give an account of non-deterministic systems and is a natural quantification for systems with choice. Computational methodologies which are probabilistic in nature are frequently more efficient and natural than conventional programming methods. We wish to produce a calculus for reasoning about such systems.

We hope that by giving an account of *relative frequency* of choice we can address some of the issues of fairness in concurrent systems. In particular we will be looking at the choice operator of our calculus. Conventionally, we write a choice as

$$A + B$$

and interpret it as a process that has the capabilities of both the processes A and B. This choice is underspecified. In an unbiased environment - one which lets the process decide which choice to take - we could see behaviours ranging from infinite selection of one process over the other to some fixed ratio of choice between the processes.

There is a large body of theory about Markov chains [Kei]. In our study of probabilistic concurrent systems, we will be able to describe our processes as Markov systems and then use the extant techniques

Our calculus is synchronous, since we shall be quantifying the relative frequency of free simultaneous choice. In an asynchronous system choices may be resolved at arbitrary times, thus giving a choice which may not be between equally free objects. This calculus simplifies the probabilistic system of [SST89,GSST90] and is based upon Milner's [Mil83] *synchronous calculus of communicating systems*. We choose not to quantify our choices directly with probabilities but to use *weights* which we can interpret as probabilistic specification via the concept of relative frequency; our notion of equality will allow us to make that interpretation. Our intention is that processes such as the following will be identified:

$$1P \ + \ 2Q \text{ and } 3P \ + \ 6Q.$$

That is to say that we shall see 1 occurrence of P for every 2 of Q in the left hand process, and shall see 3 occurrences of P to every 6 of Q in the right hand process. Since the relative frequency of P and Q is the same in both systems we would like to equate these process descriptions.

Weights are operationally more convenient than probabilities since we do not have to ensure that they normalize to any particular value. This normalization problem is the cause of the complexity of the derivation law for permission in probabilistic systems. Using weights the permission rule is literally the pruning of sub-processes that will never execute as none of their actions are permitted.

As in [SST89,GSST90,Tof89,Tof90] we shall work with a multi-coloured transition system. That is to say we have more than one notion of evolution. We intend to use two different mechanisms of state change, one being a weighted choice, the other computation. This method is used because it makes the underlying mechanism of state change clear and to obtain a rich system which is capable of expressing most of the processes we wish to study. In addition we have a stratified model [GSST90] which is suspected to be the most abstract possible for such systems.

In this paper we define the calculus WSCCS, *a weighted synchronous calculus of communicating systems.* We will define the natural relative equality via the notion of a direct equality, and show that both of these equalities are substitutive. We introduce a special weight which denotes dominance and show that it is consistent with our calculus. Expressions using this weight can be interpreted as giving processes a priority structure and we give an interpretation operator in the style of [BBK86]. We present an axiom system and demonstrate that it is sound and complete for finite process terms.

2 The Language WSCCS.

Our language WSCCS is an extension of Milner's SCCS [Mil83]. To define our language we presuppose an abelian group Act of atomic action symbols with identity $\mathbf{1}$ and the inverse of a being $\overline{a}$. As in SCCS, the complements a and $\overline{a}$ form the basis of communication. We also take a set of weights $\mathcal{W}$, denoted by w_i, which are the positive natural numbers $\mathcal{P}$ and a set of process variables Var.

The collection of WSCCS expressions ranged over by E is defined by the following BNF expressions, where $a \in Act$, $X \in Var$, $w_i \in \mathcal{W}$, S ranging over renaming functions, those $S : Act \longrightarrow Act$ such that $S(\mathbf{1}) = \mathbf{1}$ and $S(\overline{a}) = S(\overline{a})$, action sets $A \subseteq Act$, with $\mathbf{1} \in A$, and arbitrary *finite* indexing sets I:

$$E ::= X \mid a.E \mid \Sigma_{i \in I} w_i E_i \mid E \times E \mid E[A \mid E[S] \mid \mu_i \tilde{x} \tilde{E}.$$

We let Pr denote the set of closed expressions, and add $\mathbf{0}$ to our syntax, which is defined by $\mathbf{0} \stackrel{def}{=} \Sigma_{i \in \emptyset} w_i E_i$.

The informal interpretation of our operators is as follows:

- $\mathbf{0}$ a process which cannot proceed;
- X the process bound to the variable X;
- $a.E$ a process which can perform the action a whereby becoming the process described by E;

- $\Sigma_{i\in I} w_i E_i$ the *weighted* choice between the processes E_i, the weight of the outcome E_i being determined by w_i. We think in terms of repeated experiments on this process and we expect to see over a large number of experments the process E_i being chosen with a relative frequency of $\frac{w_i}{\Sigma_{i\in I} w_i}$.
- $E \times F$ the synchronous parallel composition of the two processes E and F. At each step each process must perform an action, the composition performing the composition (in Act) of the individual actions;
- $E\lceil A$ represents a process where we only permit actions in the set A. This operator is used to enforce communication and bound the scope of actions;
- $E[S]$ represents the process E relabelled by the function S;
- $\mu_i \tilde{x}\tilde{E}$ represents the solution x_i taken from solutions to the mutually recursive equations $\tilde{x} = \tilde{E}$.

Often we shall omit the dot when applying prefix operators; also we drop trailing $\mathbf{0}$, and will use a binary plus instead of the two (or more) element indexed sum, thus writing $\Sigma_{1,2}\{1_1 a.\mathbf{0},\ 2_2 b.\mathbf{0}\}$ as $1a + 2b$. Finally we allow ourselves to specify processes definitionally, by providing recursive definitions of processes. For example, we write $A \stackrel{def}{=} a.A$ rather than $A \stackrel{def}{=} \mu x.ax$. We now give some example processes.

2.1 Gambler's Ruin.

We take the actions *1*, *2*, *3*, *4*, *5*, *6* and describe a fair die as the following process:

$$Die \stackrel{def}{=} 1\overline{1}.Die + 1\overline{2}.Die + 1\overline{3}.Die + 1\overline{4}.Die\ 1\overline{5}.Die + 1\overline{6}.Die,$$

and a gambler with an amount of money n, who wins 6 units of money when a six is thrown and loses his stake (of size one) on any other outcome as the following process (we add the actions *win* and *lose*):

$$\begin{aligned} Gamb(0) &\stackrel{def}{=} \mathbf{0} \\ Gamb(n) &= 11lose.Gamb(n-1) + 12lose.Gamb(n-1) \\ &\quad\ 13lose.Gamb(n-1) + 14lose.Gamb(n-1) \\ &\quad\ 15lose.Gamb(n-1) + 16win.Gamb(n+6) \end{aligned}$$

The process

$$Game(n) \stackrel{def}{=} (Gamb(n) \times Die)\lceil\{\mathbf{1},\ win,\ lose\}$$

describes a betting game in which the gambler starts with n units of money. We can use this description to find the Markov system that the above process describes and we could predicate such parameters as the expected time until our gambler is bankrupt. We can make our die unfair by changing the weighting of the various outcomes, such an unfair die can be described as,

$$Die \stackrel{def}{=} 21.Die + 12.Die + 13.Die + 14.Die + 25.Die + 16.Die;$$

this die only produces a six one time in eight and thus our gambler would run out of money sooner. If in the group Act the operation on number actions is addition, then the process

$$Die \times Die$$

describes rolling two die and adding their outcomes.

2.2 Spontaneous Failure.

Most systems are subject to spontaneous failure due to hardware faults or disruption of electricity supply; but most descriptions of systems assume that they do not have behaviour of this nature. The process

$$Fail \stackrel{def}{=} m1.Fail + 1fail.\mathbf{0}$$

describes a system that emits the action *fail* with weight 1 at each turn and then becomes the nil process. If we take an idealized process M describing a system without spontaneous failure then the behaviour of the process

$$M \times Fail$$

will be that of M except on a *fail* action where the process will subsequently stop. Note that there is no communication structure allowing M to prevent the occurrence of a fail or even influence its likelihood.

Whilst a calculus built from natural weights may at first seem unnatural as a way of expressing probabilities, we can consider the calculus as describing rational probabilities, which we have forced into naturals by multiplication by a suitable factor. We shall see later that multiplication of weights by a constant factor is conserved by our interpretation of WSCCS processes.

3 The Semantics of WSCCS.

In this section we define the operational semantics of WSCCS. The semantics is transition based, structurally presented in the style of [Plo81], defining the actions that a process can perform and the weight with which a state can be reached. In Figure 1 we present the operational rules of WSCCS. They are presented in a natural deduction style. The transitional semantics of WSCCS is given by the least relation $\longrightarrow \subseteq WSCCS \times Act \times WSCCS$ and the least multi-relation $\longmapsto \subseteq bag(WSCCS \times \mathcal{W} \times WSCCS)$ [1], which are written $E \xrightarrow{a} F$ and $E \stackrel{w}{\longmapsto} F$ respectively, satisfying the rules laid out in Figure 1. These rules respect the informal description of the operators given earlier. The reason that process are multi-related by weight is that we may specify more than one way to choose the same process with the same weight and we have to retain all the copies. For example, the process

$$1P + 1P + 1Q$$

can evolve to the process P with commulative weight 2, so that we have to retain both evolutions.

The predicate $does_A(E)$ is well defined since we have only permitted finitely branching choice expressions. The action of the permission operator is to prune from the choice tree those processes that can no longer perform any action.

3.1 Direct Bisimulation.

Our bisimulations will be based on the accumulation technique of Larsen and Skou [LS89]. We start by defining accumulations of evolutions for both types of transition.

[1] Where $\longmapsto \subseteq Bag(WSCCS \times \mathcal{W} \times WSCCS)$ is the bag whose elements are those of the set $WSCCS \times \mathcal{W} \times WSCCS$, with the usual notion of bag.

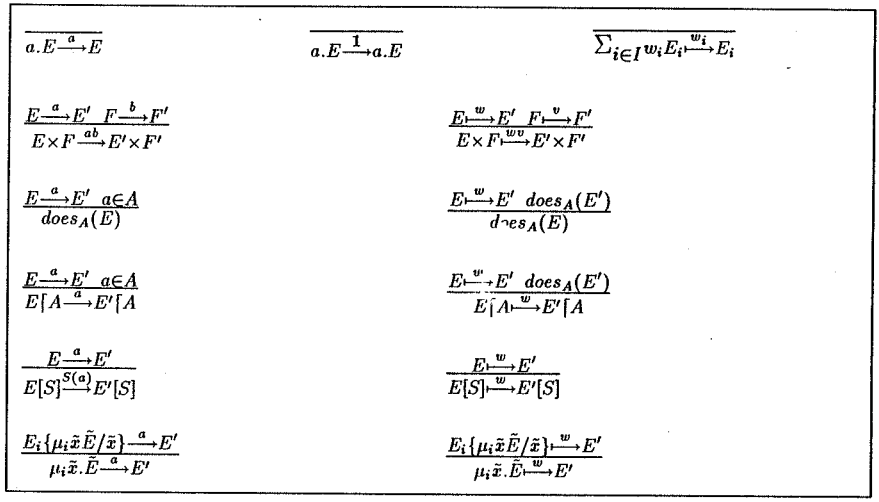

Figure 1: Operational Rules for WSCCS.

Definition 3.1 *Let S be a set of processes then:*

- $P \xmapsto{w} S$ *iff for all* $Q \in S$ *and* w_i *such that* $P \xmapsto{w_i} Q$[2] *then* $w = \Sigma w_i$;
- $P \xrightarrow{a} S$ *iff there exists* $Q \in S$ *and* $P \xrightarrow{a} Q$.

We define a form of bisimulation that identifies two processes if the total weight of evolving into any equivalent states is the same. This is not quite the indentification we wish to make, but we will make such an identification later.

Definition 3.2 *An equivalence relation* $R \subseteq Pr \times Pr$[3] *is a* direct bisimulation *if* $(P,Q) \in R$ *implies for all* $S \in Pr/R$ *that:*

- *for all* $w \in \mathcal{W}$, $P \xmapsto{w} S$ *iff* $Q \xmapsto{w} S$;
- *for all* $a \in Act$, $P \xrightarrow{a} S$ *iff* $Q \xrightarrow{a} S$.

Two processes are direct bisimulation equivalent, *written* $P \overset{d}{\sim} Q$, *if there exists a direct bisimulation R between them.*

Definition 3.3

$$\overset{d}{\sim} \equiv \bigcup\{R \mid R \text{ is a direct bisimulation }\}.$$

[2] Remembering this is a multi-relation so some of the Q and w_i may be the same process and value. We take all occurences of processes in S and add together all the weight arrows leading to them.

[3] We denote the equivalence class of a processes P with respect to R by $[P]_R$. When it is clear from the context to which equivalence we are refering, we will omit the subscript.

That $\overset{d}{\sim}$ is an equivalence follows immediately from it being a union of equivalences.

Lemma 3.4 *Let P and Q be processes such that $P \overset{d}{\sim} Q$. Then for all action sets A, $does_A(P)$ iff $does_A(Q)$.*

Proof: Immediate from the definition of direct bisimulation.

Proposition 3.5 *Direct equivalence is substitutive for finite processes. Thus, given $P \overset{d}{\sim} Q$ and $P_i \overset{d}{\sim} Q_i$ for all $i \in I$ then:*

1. $a.P \overset{d}{\sim} a.Q$;
2. $\Sigma_{i\in I} w_i P_i \overset{d}{\sim} \Sigma_{i\in I} w_i Q_i$;
3. $P \times E \overset{d}{\sim} Q \times E$;
4. $P\lceil A \overset{d}{\sim} Q\lceil A$;
5. $P[S] \overset{d}{\sim} Q[S]$.

Proof:(After [Mil89, GSST90]) We proceed by induction over the structure of the process and the size of the process:

1. $a.P \overset{d}{\sim} a.Q$ immediate;
2. $\Sigma_{i\in I} w_i P_i \overset{d}{\sim} \Sigma_{i\in I} w_i Q_i$ immediate;
3. $P \times E \overset{d}{\sim} Q \times E$, we construct the equivalence relation $R = \{(P \times E, Q \times E) \mid P \overset{d}{\sim} Q\} \cup \{(P \times E, E \times Q) \mid P \overset{d}{\sim} Q\}$. So consider the weight evolutions of the two processes,

   if $P \times E \overset{w}{\longmapsto} [P' \times E']$ then
   $P \overset{w_1}{\longmapsto} [P']$ and $E \overset{v_1}{\longmapsto} [E']$ also (possibly)
   $P \overset{w_2}{\longmapsto} [E']$ and $E \overset{v_2}{\longmapsto} [P']$
   with $w = w_1 v_1 + w_2 v_2$, hence
   $Q \overset{w_1}{\longmapsto} [P']$ and
   $Q \overset{w_2}{\longmapsto} [E']$, thus
   $Q \times E \overset{w}{\longmapsto} [Q' \times E'] = [P' \times E']$.

   Likewise for the action evolutions,

   if $P \times E \xrightarrow{a} [P' \times E']$ then
   $P \xrightarrow{b} [P']$ and $E \xrightarrow{c} [E']$
   with $a = bc$,
   thus $Q \xrightarrow{b} [P']$ hence
   $Q \times E \xrightarrow{a} [Q' \times E'] = [P' \times E']$.

   The symmetric cases follow immediately by the same proof;
4. $P\lceil A \overset{d}{\sim} Q\lceil A$ we define $R \equiv \{(P\lceil A, Q\lceil A) \mid P \overset{d}{\sim} Q\}$. Now consider the weight evolutions of the processes,

   if $P\lceil A \overset{w}{\longmapsto} [P'\lceil A]$ then
   $P \overset{w}{\longmapsto} [P']$, and hence
   $Q \overset{w}{\longmapsto} [P']$, thus

$Q\lceil A \xrightarrow{w} [Q'\lceil A]$.

From lemma 3.4 as $does_A(Q')$ iff $does_A(P')$ since $Q' \overset{d}{\sim} P'$,
so $[Q'\lceil A] = [P'\lceil A]$ as required.

Similarly for the action evolutions

if $P\lceil A \xrightarrow{a} [P'\lceil A]$ with $a \in A$ then
$P \xrightarrow{a} [P']$, and hence
$Q \xrightarrow{a} [P']$, thus
$Q\lceil A \xrightarrow{w} [Q'\lceil A] = [P'\lceil A]$;

5. $P[S] \overset{d}{\sim} Q[S]$ immediate.

We proceed by the usual technique of pointwise extension to define our equivalence for infinite processes.

Definition 3.6 *Let $\tilde{E}$ and $\tilde{F}$ be expressions containing variables at most $\tilde{X}$. Then we will say $\tilde{E} \overset{d}{\sim} \tilde{F}$ if for all process sets $\tilde{P}$, $\tilde{E}\{\tilde{P}/\tilde{X}\} \overset{d}{\sim} \tilde{F}\{\tilde{P}/\tilde{X}\}$.*

Proposition 3.7 *If $\tilde{E} \overset{d}{\sim} \tilde{F}$ then $\mu_i\tilde{X}.\tilde{E} \overset{d}{\sim} \mu_i\tilde{X}.\tilde{F}$.*

Proof: As for Proposition 3.5 but inducing over the depth of inference required to infer either a weight evolution or an action evolution.

3.2 Relative Bisimulation.

Unfortunately, the congruence given by direct bisimulation is too strong; it does not capture our notion of relative frequency, but captures total frequency. Since we would like to be able to equate processes such as,

$$2P + 3Q \text{ and } 4P + 6Q,$$

we need to weaken our notion of equality. The basic idea is that in order to show two processes equivalent, for each pair of equivalent states we can choose a *constant* factor such that the total weight of equivalent immediate derivatives is related by multiplication by that factor. If we can do this for all potentially equivalent states then we will say that the processes are the same in terms of relative frequency. Since the constant factor may well need to be a rational (and we wish to keep our numbers as simple as possible) we will actually use two constants in comparing relative frequency. This allows us to use a symmetrical defintion.

Definition 3.8 *We say an equivalence relation $R \subseteq Pr \times Pr$ is a* relative bisimulation *(its equivalence classes being Pr/R) if $(P,Q) \in R$ implies that:*

1. *There are $c_1, c_2 \in \mathcal{P}$ such that for all $S \in Pr/R$ and for all $w \in \mathcal{W}$; $P \xrightarrow{w} S$ iff $Q \xrightarrow{v} S$ and $c_1 w = c_2 v$,*
2. *For all $S \in Pr/R$ and for all $a \in Act$, $P \xrightarrow{a} S$ iff $Q \xrightarrow{a} S$;*

two processes are relative bisimular equivalent, *written* $P \overset{r}{\sim} Q$ *if there exists a relative bisimulation* R *between them.*

We have chosen to use multiplication by a constant rather than division as this permits us to stay within the natural numbers. We could have normalized so that the total weight actions of any state is 1, and then we would have had an equivalence that is identical to that of stratified bisimulation [SST89,GSST90].

Definition 3.9

$$\overset{r}{\sim} \equiv \bigcup\{R \mid R \text{ is a relative bisimulation}\}.$$

Proposition 3.10 *Let* P *and* Q *be processes such that* $P \overset{d}{\sim} Q$, *then* $P \overset{r}{\sim} Q$.

Proof: Immediate from the definitions of $\overset{d}{\sim}$ and $\overset{r}{\sim}$.

Definition 3.11 *Let* $\tilde{E}$ *and* $\tilde{F}$ *be expressions containing variables at most* $\tilde{X}$. *Then we will say* $\tilde{E} \overset{r}{\sim} \tilde{F}$ *if for all process sets* $\tilde{P}$, $\tilde{E}\{\tilde{P}/\tilde{X}\} \overset{r}{\sim} \tilde{F}\{\tilde{P}/\tilde{X}\}$.

Proposition 3.12 $\overset{r}{\sim}$ *is a congruence for finite and infinite processes.*

Proof: This is identical to that of Proposition 3.5 excepting the normalisation factor, but the factor is conserved by the operations of WSCCS. Thus the congruence property follows directly from $\overset{d}{\sim}$ being a congruence.

Whilst we could axiomatize this system and these notions of equality we prefer to wait until after our consideration of the addition of dominance in the next section.

4 Adding Infinite Frequency.

The notion of priority, the arbitrary choice of one possibility over another, is a natural control mechanism for computational machinery. The paper [CH88] provides an operational semantics for a single level of priority, and in [BBK86] an equational formulation is given for arbitrary priority over actions.

In this system we will represent priority as the dominance of one choice of future state over another, in other words, in terms of processes. This is interpreted as having a state whose relative frequency is infinite with respect to other states into which it can evolve. If we were to experiment on such a process, in a free context, we would only ever see occurrences of the dominant state.

There are two approaches to introducing this concept to our system of weights: a zero weight or an infinite weight. There are

$$0P + 3Q \text{ and } 1P + \omega Q,$$

both being natural candidates for a process where the state Q has priority over the state P. We choose to use the latter form as it matches more closely our notion of dominance, or infinite frequency. Thus we extend our weight set $\mathcal{W}$ to be $\mathcal{P} \cup \{\omega\}$ with the following multiplication and addition rules for ω:

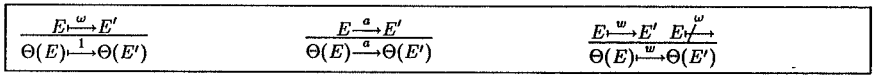

Figure 2: The priority interpretation operator Θ.

$$\omega + n = n + \omega = \omega + \omega = \omega,$$
$$\omega * n = n * \omega = \omega * \omega = \omega.$$

Note that addition and multiplication are still associative and commutative in the extended weight set.

Proposition 4.1 $\overset{d}{\sim}$ *is a congruence for processes defined over the augmented weight set.*

> **Proof:** We simply take account of the possibility of ω weighted states, and since they are distinct and stay distinct under composition then the equivalence will be substitutive.

For the relative case we have to be a little more careful. In the definition of $\overset{r}{\sim}$ we are only allowed to use normalizing factors that are in $\mathcal{P}$. If we weaken this to $\mathcal{W}$, in the case of our augmented weight set, then we have that for $n, m, r, s \in \mathcal{P}$:

$$nP + mQ \sim \omega P + \omega Q \sim rP + sQ,$$

but not necessarily that

$$nP + mQ \sim rP + sQ,$$

contradicting the requirement that our relation be an equivalence.

Proposition 4.2 $\overset{r}{\sim}$ *is a congruence for processes over the augmented weight set.*

> **Proof:** A trivial extension of the proof of Proposition 4.1.

The result of not allowing factors from the whole of $\mathcal{W}$ is that the following two processes cannot be identified:

$$nP \text{ and } \omega P.$$

More generally, we should like to say, in a free context, that the following processes are identical:

$$\omega P + nQ + mR \text{ and } 1P.$$

In order to enable us to make the comparison and to interpret our priority information, we define a priority interpretation operator Θ in the style of [BBK86]. Its operational rules are given in Figure 2. It should be noted that since we have restricted ourselves to finitely branching processes the negated evolution in the definition of Θ is decidable.

Proposition 4.3 *Let P and Q be processes such that $P \overset{d}{\sim} Q$, then $\Theta(P) \overset{d}{\sim} \Theta(Q)$.*

> **Proof:** Follows from independence of ω weights to the presence of other weights.

Proposition 4.4 *Let P and Q be processes such that $P \stackrel{\Sigma}{\sim} Q$, then $\Theta(P) \stackrel{\Sigma}{\sim} \Theta(Q)$.*

Proof: A trivial extension to Proposition 4.3.

The following demonstrates the expressive power of our notion of priority.

Definition 4.5 *Let $\mathcal{T}$ be an* n-ary tree *of processes, with $P^r_{\mathcal{T}}$ is root, and sub-trees $\mathcal{T}_1, \ldots, \mathcal{T}_n$, then the process $P_{\mathcal{T}}$ respects a process with the priority structure given by the tree. We define $P_{\mathcal{T}}$ recursively over the structure of the tree:*

$$P_{\mathcal{T}} \stackrel{def}{=} \omega P^r_{\mathcal{T}} + 1P_{\mathcal{T}_1} + \ldots + 1P_{\mathcal{T}_n}.$$

Proposition 4.6 *Let $\mathcal{T}$ be a tree of processes with $P^r_{\mathcal{T}}$ its root, then $\Theta(P_{\mathcal{T}}) \stackrel{\Sigma}{\sim} 1P^r_{\mathcal{T}}$*

Proof: Immediate from the definitions.

We now present some example uses of the priority structure.

4.1 Broadcasting.

In some systems, such as distributed system of machines each of which can fail, it is necessary to be able to send a signal to as many processes as will listen. This we can represent as a bias towards sending as many copies as possible of an action. Let us say we wish to broadcast $\overline{a}$. Then we can define a process which transmits to up to n other processes:

$$Brd(n) = \omega\overline{a}^n.Brd(n) + 1(\omega\overline{a}^{n-1}.Brd(n) + 1(\ldots + \mathbf{1}\mathbf{1}.Brd(n))).$$

4.2 A logon controller.

We wish to define a system with the following properties:

- There are three types of user p, s and l;
- Logoffs are always dealt with first;
- When they are fewer than five users no one has priority;
- When there are five to ten users then those logging on as p or l have priority to those logging on as s;
- When there are more than ten users then p will have priority over s and l, and l will have priority over s.

We call our process *Log* and paramaterize it by the number of users.

$$\begin{aligned}
Log(0) &= \omega(1p.Log(n+1) + 1s.Log(n+1) + 1l.Log(n+1)) + \mathbf{1}\mathbf{1}.Log(n)\\
Log(1 \leq n \leq 5) &= \omega off.Log(n-1) + 1(\omega(1p.Log(n+1) + 1s.Log(n+1) +\\
&\qquad 1l.Log(n+1)) + \mathbf{1}\mathbf{1}.Log(n))\\
Log(6 \leq n \leq 10) &= \omega off.Log(n-1) + 1(\omega(1p.Log(n+1) + 1l.Log(n+1)) +\\
&\qquad 1(\omega s.Log(n+1) + \mathbf{1}\mathbf{1}.Log(n)))\\
Log(11 \leq n) &= \omega off.Log(n-1) + 1(\omega p.Log(n+1) + 1(\omega l.Log(n+1) +\\
&\qquad 1(\omega s.Log(n+1) + \mathbf{1}\mathbf{1}.Log(n))))
\end{aligned}$$

$$(\Sigma_1)\ \Sigma_{i\in I} w_i E_i = \Sigma_{j\in J} v_j E_j \quad \begin{cases} \text{there is a surjection } f : I \longmapsto J \text{ with} \\ v_j = \Sigma\{w_i | i \in I \wedge f(i) = j\}, \text{ and for all } i \text{ with } f(i) = j \text{ then} \\ E_i = E_j. \end{cases}$$

Figure 3: Equational rules for WSCCS_0.

5 Equational Theory.

We develop a set of equational laws for WSCCS, in a similar fashion to [CJS90,MT89], which are sound with respect to $\overset{d}{\sim}$ defined above. We shall use $=$ to represent derivability in out equational theory and $\equiv$ to represent syntactic identity. We postpone the development of an equational system for $\overset{\tau}{\sim}$ until the end of this section.

As a start we restrict our attention to the finite subset given by the following syntax:

$$E ::= a.E \mid \Sigma_{i\in I} w_i E_i,$$

and we refer to this language as WSCCS_0.

Our equational theory is presented in Figure 3. We can straightforwardly demonstrate that these laws are sound with respect to $\overset{d}{\sim}$. Hence:

Proposition 5.1 $p = q \implies p \overset{d}{\sim} q$

Definition 5.2 *We define when an expression is in normal form.*

- *A term is in* **NF** *if it is in* **pNF** *or* **ΣNF** .
- *A term is in* **pNF** *if it is of the form* $a.$**NF** .
- *A term is in* **ΣNF** *if it is of the form* $\Sigma_{i\in I} w_i E_i$ *with* $E_i \neq E_{i'}$ *for* $i \neq i'$ *and all* E_i *in* **NF** .

Definition 5.3 *The* stratification depth *of a process, written* $s(E)$ *is defined recursively as follows:*

$$s(a.E) = 0; \quad s(\Sigma_{i\in I} w_i E_i) = 1 + max\{s(E_i)\}.$$

Proposition 5.4 *Every term in WSSCS*$_0$ *can be equated to a term in* **NF** *using the laws of Figure 3.*

Proof:

- $a.E$ is in **pNF** .
- $\Sigma_{i\in I} w_i E_i$ either is in **ΣNF** , or can be reduced to it by one application of the law Σ_1. By indentifying a set of distinct processes call it J and since $J \subseteq I$ we can arrange that the weights v_j are chosen suitably, thus we have a map $f : I \longmapsto J$ which has the required properties.

We now demonstrate that our normal forms are distinct.

Proposition 5.5

(Exp_1) $a.E \times b.F = ab.(E \times F)$ $\qquad$ (Exp_2) $a.E \times \Sigma_{j\in J} v_j F_j = \Sigma_{j\in J} v_j (a.E \times F_j)$

(Exp_3) $\Sigma_{i\in I} w_i E_i \times \Sigma_{j\in J} w_j F_j = \Sigma_{(i,j)\in(I\times J)} w_i v_j (E_i \times F_j)$

Figure 4: Expansion theorems.

- *If* $P \overset{d}{\sim} Q$ *and* P *is in* **pNF** *then* Q *is in* **pNF** .
- *If* $P \overset{d}{\sim} Q$ *and* P *is in* Σ**NF** *then* Q *is in* Σ**NF** .

Proof: Directly from the fact that a **pNF** can perform an immediate computation action, whilst a Σ**NF** cannot.

Proposition 5.6

1. *If* $P \overset{d}{\sim} Q$ *and both* P *and* Q *are in* **pNF** *then* $P = Q$.
2. *If* $P \overset{d}{\sim} Q$ *and both* P *and* Q *are in* Σ**NF** *then* $P = Q$.

Proof: Induction over the size of P and Q.

1. Immediate from congruence properties of $\overset{d}{\sim}$.
2. We know the first evolution must be a weight evolution. Since P and Q are in Σ**NF** consider that $P \equiv \Sigma_{i\in I} w_i E_i$ and $Q \equiv \Sigma_{j\in J} v_j E_j$ now we know that $P \overset{w}{\longmapsto} [E_i]$ iff $Q \overset{w}{\longmapsto} [E_j]$ and that all the E_i and E_j are distinct. Thus for each w_i and E_i there is a unique matching v_j and E_j, and vice-versa. Hence we can apply Σ_1 to derive the equality, as we have a bijection between I and J with the requisite properties.

We extend our language from WSCCS_0 to WSCCS_1 by adding the operator $\times$. The expansion laws needed to axiomatize $\times$ are presented in Figure 4; as they can be derived, the laws for commutativity and associativity have been omitted.

Proposition 5.7 *For any two NF terms* p *and* q *there is a term* r *not involving* $\times$ *such that* $p \times q = r$.

Proof: By induction over the number of occurrences of $\times$ and the stratification depth of p and q.

Further we extend from WSCCS_1 to WSCCS_2 by adding the permission operator. We need to axiomatize the predicate $does_A(E)$ in order to give a complete axiomatic presentation. The laws of Figure 5 axiomatize permission.

Definition 5.8 *Let* A *be an action set then the predicate,* $d_A(E)$*, expressing the fact that* E *can perform an action in* A*, is defined recursively as follows:*

- *If* $a \in A$ *then* $d_A(a.E)$;

(Res_1) $(a.E)\lceil A = \begin{cases} a.(E\lceil A) \text{ if } a \in A \\ \mathbf{0} \text{ otherwise.} \end{cases}$

(Res_2) $(\Sigma_{i\in I} w_i E_i)\lceil A = \Sigma_{j\in J} w_j (E_j \lceil A)$ where $J = \{i \in I | d_A(E_i)\}$

Figure 5: Axiomatization of permission.

(Θ_1) $\Theta(a.E) = a.\Theta(E)$

(Θ_2) $\Theta(\Sigma_{i\in I} w_i E_i) = \begin{cases} \Sigma_{j\in J} 1.\Theta(E_j) \text{ where } J = \{i \in I | w_i = \omega\} \text{ and } J \neq \emptyset, \\ \Sigma_{i\in I} w_i \Theta(E_i) \text{ if } J = \emptyset \end{cases}$

Figure 6: Priority Axiomatization.

- *If there exists $i \in I$ with $d_A(E_i)$ then $d_A(\Sigma_{i\in I} w_i E_i)$.*

Proposition 5.9 *Let E be in* **NF** *then $d_A(E)$ iff $does_A(E)$*

Proof: Immediate from the definitions.

Proposition 5.10 *For any* **NF** *term p there is an* **NF** *term r not using the permission operator such that $p\lceil A = r$.*

Proof: By induction over the number of permission operators in p and the stratification depth of p.

Finally we extend from $WSCCS_2$ to $WSCCS_3$ by adding the priority interpretation operator Θ. The laws required are presented in Figure 6.

Proposition 5.11 *Let p be a term in* **NF** *then there is a term r not involving Θ such that $r = \Theta(p)$.*

Proof: By induction over the number of operators Θ and the stratification depth of p.

Proposition 5.12 *(Completeness) Our axiom system Σ_1, Exp_1, Exp_2, Exp_3 , Res_1, Res_2, Θ_1, Θ_2 is complete for the language $WSCCS_3$.*

Proof: Directly from Propositions 5.11, 5.10, 5.9, 5.7, 5.6, 5.5 and 5.4.

We have axiomatized $\overset{d}{\sim}$ and not $\overset{\tau}{\sim}$; we can add the axiom of Figure 7. Along with the earlier axioms this gives a complete and sound axiomatization of WSCCS with respect to $\overset{\tau}{\sim}$.

Proposition 5.13 *Let $p \overset{\tau}{\sim} q$ with p and q in Σ***NF** *then $p = q$.*

Proof: As in the direct case but we may need an application of *Ren*.

(*Ren*) $\Sigma_{i\in I} w_i E_i = \Sigma_{i\in I} n w_i E_i$ where $n \in \mathcal{P}$

Figure 7: Renormalization axiom.

6 Value Passing.

In order to obtain processes where the relative frequencies may be dynamic we introduce a values passing version of WSCCS. This is defined in the same manner as the value passing extension to CCS [Mil89]. However our value domain has to be finite, as we are not allowed expressions with infinite branching. We take the following syntax:

- Values in $\mathcal{V}$;
- Expressions $e_1, e_2, \ldots$;
- Weight expressions $[e]$;
- Prefixes $\lambda(x).E$, $\overline{\lambda}(e).E$, $\mathbf{1}.E$;
- Summation $\Sigma_{i\in I}[e_i]E_i$;
- Product $E \times F$;
- Permission $E\lceil A$;
- Renaming $E[S]$;
- Constants $A(x_1, \ldots, x_n) \stackrel{def}{=} E$ with the free variables of E contained in $\tilde{x}$.

We translate the constant $A(x_1, \ldots, x_n) \stackrel{def}{=} E$ as the set of defining equations $\{A_{e_1,\ldots,e_n} \stackrel{def}{=} \hat{E}; e_1, \ldots e_n \in \mathcal{V}\}$.

Definition 6.1 *We define the translation of an expression E in the value passing calculus to an expression $\hat{E}$ in WSCCS, recursively over the structure of E:*

E	$\hat{E}$
$\lambda(x).E$	$\Sigma_{x\in\mathcal{V}} 1\lambda_x.\hat{E}$
$\overline{\lambda}(e).E$	$\overline{\lambda_e}.\hat{E}$
$\mathbf{1}.E$	$\mathbf{1}.\hat{E}$
$\Sigma_{i\in I}[e_i]E_i$	$\Sigma_{i\in I}[e_i]\hat{E}_i$
$E \times F$	$\hat{E} \times \hat{F}$
$E[S]$	$\hat{E}[\hat{S}]$ *where* $S(\hat{\lambda}_x) = S(\lambda)_x$
$E\lceil A$	$\hat{E}\lceil\{\mathbf{1}\} \cup \{\lambda_x \mid \lambda \in A, x \in \mathcal{V}\}$
$A(e_1, \ldots, e_n)$	$A_{e_1,\ldots,e_n}$

There are several examples of the use of this system, but they are too large to include and will be presented separately.

7 Conclusions and further work.

We have presented a calculus of relative frequency that is both elegant and simple. We can express any degree of relative frequency between processes; for example:

$$nA + mB \qquad \omega A + nB \qquad \omega A + \omega B$$

express respectively that A and B occur, in a fixed ratio, with A dominant, and in any ratio. The notion of priority developed is very flexible and gives us a high degree of expressive power. Unlike the work of [BBK86] as we have encoded priority into our processes it is dynamic with respect to particular actions.

Currently the author is using this calculus to represent the behaviour of communal insects, a study in concurrency in the large. We would like to extend our axiom system to finite state expressions as in [CJS90]. Although WSCCS is elegant, in order to express some of the approximate equalities of [CJS90] we would need some complex normalizations encoded into our equality.

8 Acknowledgements.

I would like to thank Joachim Parrow and David Pym for some very helpful discussions on the content of this work. The concurrency club especially David Walker, Colin Stirling and Faroe Mille for their interest. Scott Smolka and Berhard Steffen for interesting me, once again, in the problem of probabilistic concurrent computation.

9 Bibliography.

[BBK86] J. Baeten, J. Bergstra and J. Klop, Syntax and defining equations for an interrupt mechanism in process algebra, Fundamenta Informatica IX, pp 127-168, 1986

[CH88] R. Cleveland and M. Hennessey, Priorites in Process Algebras, proceedings LICS 1988.

[GJS90] A. Giaclone, C.-C. Jou, and S. A. Smolka, Algebraic reasoning for probabilistic concurrent systems, proceedings of working conference on programming concepts and methods IFIP TC 2, 1990.

[GSST90] R. van Glabbek, S. A. Smolka, B. Steffen and C.Tofts, Reactive, Generative and Stratified Models of Probabilistic Processes, proceedings LICS 1990.

[JP89] C. Jones and G. D. Plotkin, A probabilistic powerdomain of evaluations, proceedings LICS 1989.

[Kei] J. Keilson, Markov Chain Models - Rarity and exponentiality, Applied Mathematical Sciences 28, Springer Verlag.

[LS89] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. proceedings POPL 1989.

[Mil83] R. Milner, Calculi for Synchrony and Asynchrony, Theoretical Computer Science 25(3), pp 267-310, 1983.

[Mil89] R. Milner, Communication and Concurrency, Prentice Hall, 1989.

[MT89] F. Moller and C.Tofts, a Temporal Calculus of Communicating Systems, LFCS-89-104, University of Edinburgh.

[Par81] D. Park, Concurrency and Automata on infinite sequences, Springer LNCS 104.

[Plo81] G. D. Plotkin, A structured approach to operational semantics. Technical report Daimi Fn-19, Computer Science Department, Aarhus University. 1981

[Rab76] M. Rabin, Probabilistic algorithmns, Algorithmns and Complexity, recent results and new directions, J. Traub ed Academic press, New York, 1976 pp 21-39.

[SST89] S. Smolka, B. Steffen and C. Tofts, unpublished notes. Working title, Probability + Restriction $\Rightarrow$ priority.

[Tof89] C. Tofts, Timing Concurrent Processes, LFCS-89-103, University of Edinburgh.

[Tof90] C. Tofts, Proof Methods and Pragmatics for Parallel Programming, Thesis in examination.

[Wel83] D. Welsh, Randomised algorithmns, Discrete applied mathematics 5, 1983 pp 133-145.

Copyright © 1990, Laboratory for Foundations of Computer Science,
University of Edinburgh. All rights reserved.

Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.